A Discriminative Learning Model for Coordinate Conjunctions

Masashi Shimbo*
Graduate School of Information Science
Nara Institute of Science and Technology
Ikoma, Nara 630-0192, Japan
shimbo@is.naist.jp

Kazuo Hara*
Graduate School of Information Science
Nara Institute of Science and Technology
Ikoma, Nara 630-0192, Japan
kazuo-h@is.naist.jp

Abstract

We propose a sequence-alignment based method for detecting and disambiguating coordinate conjunctions. In this method, averaged perceptron learning is used to adapt the substitution matrix to the training data drawn from the target language and domain. To reduce the cost of training data construction, our method accepts training examples in which complete word-by-word alignment labels are missing, but instead only the boundaries of coordinated conjuncts are marked. We report promising empirical results in detecting and disambiguating coordinated noun phrases in the GENIA corpus, despite a relatively small number of training examples and minimal features are employed.

1 Introduction

Coordination, along with prepositional phrase attachment, is a major source of syntactic ambiguity in natural language. Although only a small number of previous studies in natural language processing have dealt with coordinations, this does not mean disambiguating coordinations is easy and negligible; it still remains one of the difficulties for state-of-the-art parsers. in Charniak and Johnson's recent work (Charniak and Johnson, 2005), for instance, two of the features incorporated in their parse reranker are aimed specifically at resolving coordination ambiguities.

Previous work on coordinations includes (Agarwal and Boggess, 1992; Chantree et al., 2005; Kurohashi and Nagao, 1994; Nakov and Hearst, 2005; Okumura and Muraki, 1994; Resnik, 1999). Earlier studies (Agarwal and Boggess, 1992; Okumura and Muraki, 1994) attempted to find heuristic rules to disambiguate coordinations. More recent research are concerned with capturing structural similarity between conjuncts using thesauri and corpora (Chantree et al., 2005), or web-based statistics (Nakov and Hearst, 2005).

We identify three problems associated with the previous work.

1. Most of these studies evaluate the proposed heuristics against restricted forms of conjunctions. In some cases, they only deal with coordinations with exactly two conjuncts, leaving the generality of these heuristics unclear.

2. Most of these studies assume that the boundaries of coordinations are known in advance, which, in our opinion, is impractical.

3. The proposed heuristics and statistics capture many different aspects of coordination. However, it is not clear how they interact and how they can be combined.

To address these problems, we propose a new framework for detecting and disambiguating coordinate conjunctions. Being a discriminative learning model, it can incorporate a large number of overlapping features encoding various heuristics for coordination disambiguation. It thus provides a test bed for examining combined use of the proposed heuristics as well as new ones. As the weight on each feature is automatically tuned on the training data, assessing these weights allows us to evaluate the relative merit of individual features.

*Equal contribution.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 610–619, Prague, June 2007.

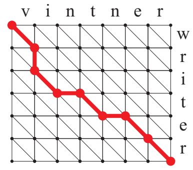

Figure 1: An alignment between 'writer' and 'vintner,' represented as a path in an edit graph

Our learning model is also designed to admit examples in which only the boundaries of coordinated conjuncts are marked, to reduce the cost of training data annotation.

The state space of our model resembles that of Kurohashi and Nagao's Japanese coordination detection method (Kurohashi and Nagao, 1994). However, they considered only the decoding of coordinated phrases and did not address automatic parameter tuning.

2 Coordination disambiguation as sequence alignment

It is widely acknowledged that coordinate conjunctions often consist of two or more conjuncts having similar syntactic constructs. Our coordination detection model also follows this observation. To detect such similar constructs, we use the sequence alignment technique (Gusfield, 1997).

2.1 Sequence alignment

Sequence alignment is defined in terms of transformation of one sequence (string) into another through an *alignment*, or a series of edit operations. Each of the edit operations has an associated cost, and the cost of an alignment is defined as the total cost of edit operations involved in the alignment. The minimum cost alignment can be computed by dynamic programming in a state space called an *edit graph*, such as illustrated in Figure 1. In this graph, a complete path starting from the upper-left initial vertex and arriving at the lower-right terminal vertex constitutes a global alignment. Likewise, a partial path corresponds to a local alignment.

Sequence alignment can also be formulated with the *scores* of edit operations instead of their *costs*. In this case, the sequence alignment problem is that of finding a series of edit operations with the maximum score.

2.2 Edit graph for coordinate conjunctions

A fundamental difference between biological local sequence alignment and coordination detection is that the former deals with finding local homologies between two (or more) distinct sequences, whereas coordination detection is concerned with local similarities within a single sentence.

The maximal local alignment between two identical sequences is a trivial (global) alignment of identity transformation (the diagonal path in an edit graph). Coordination detection thus reduces to finding *off-diagonal* partial paths with the highest similarity score. Such paths never cross the diagonal, and we can limit our search space to the upper triangular part of the edit graph, as illustrated in Figure 2.

3 Automatic parameter tuning

Given a suitable substitution matrix, i.e., function from edit operations to scores, it is straightforward to find optimal alignments, or coordinate conjunctions in our task, by running the Viterbi algorithm in an edit graph.

In computational biology, there exist established substitution matrices (e.g., PAM and BLOSUM) built on a generative model of mutations and their associated probabilities.

Such convenient substitution matrices do not exist for coordination detection. Moreover, optimal score functions are likely to vary from one domain (or language) to another. Instead of designing a specific function for a single domain, we propose a general discriminative learning model in which the score function is a linear function of the *features* assigned to vertices and edges in the state space, and the weight of the features are automatically tuned for given gold standard data (training examples) drawn from the application domain. Designing heuristic rules for coordination detection, such as those proposed in previous studies, translates to the design of suitable features in our model.

Our learning method is an extension of Collins's perceptron-based method for sequence labeling (Collins, 2002). However, a few incompatibilities exists between Collins' sequence labeling method and edit graphs used for sequence alignment.

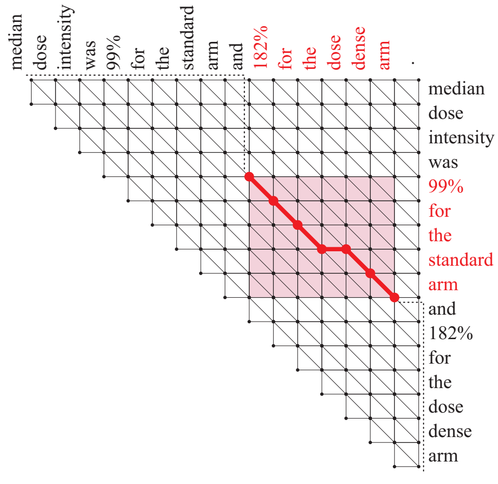

Figure 2: An edit graph for coordinate detection

1. Collins's method, like the linear-chain conditional random fields (CRFs) (Lafferty et al., 2001; Sha and Pereira, 2003), seeks for a complete path from the initial vertex to the terminal using the Viterbi algorithm. In an edit graph, on the other hand, coordinations are represented by partial paths. And we somehow need to complement the partial path to make a complete path.

2. A substitution matrix, which defines the score of edit operations, can be represented as a function of features defined on edges. But to deal with complex coordinations, a more expressive score function is sometimes desirable, so that scores can be computed not only on the basis of a single edit operation, but also on consecutive edit operations. Edit graphs are not designed to accommodate features for such a higher-order interaction of edit operations.

To reconcile these incompatibilities, we derive a more finer-grained model from the original edit graph. In presenting the description of our model below, we reserve the terminology 'vertex' and 'edge' for the original edit graph, and use 'node' and 'arc' for our new model, to avoid confusion.

3.1 State space for learning coordinate conjunctions

The new model is also based on the edit graph. In this model, we create a node for each triple (v, p, e),

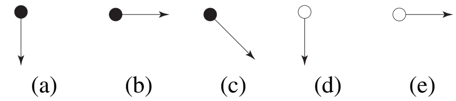

Figure 3: Five node types created for a vertex in an edit graph: (a) *Inside Delete*, (b) *Inside Insert*, (c) *Inside Substitute*, (d) *Outside Delete*, and (e) *Outside Insert*.

Figure 4: Series of edit operations with an equivalent net effect. (a) $(\mathit{Insert}, \mathit{Delete})$, and (b) $(\mathit{Delete}, \mathit{Insert})$. (b) is prohibited in our model.

where v is a vertex in the original edit graph, $e \in \{\mathit{Delete}, \mathit{Insert}, \mathit{Substitute}\}$ is an admissible[1] edit operation at v, and $p \in \{\mathit{Inside}, \mathit{Outside}\}$ is a *polarity* denoting whether or not the edit operation e is involved in an alignment.

For a node (v, p, e), we call the pair (p, e) its *type*. All five possible node types for a single vertex of an edit graph are shown in Figure 3. We disallow type $(\mathit{Outside}, \mathit{Substitute})$, as it is difficult to attribute an intuitive meaning to substitution when two words are not aligned (i.e., *Outside*).

Arcs between nodes are built according to the transitions allowed in the original edit graph. To be precise, an arc between node (v_1, p_1, e_1) and node (v_2, p_2, e_2) is created if and only if the following three conditions are met. (i) Edit operations e_1 and e_2 are admissible at v_1 and v_2, respectively; (ii) the sink of the edge for e_1 at v_1 is v_2; and (iii) it is not the case with $p_1 = p_2$ and $(e_1, e_2) = (\mathit{Delete}, \mathit{Insert})$.

Condition (iii) is introduced so as to disallow transition $(\mathit{Delete}, \mathit{Insert})$ depicted in Figure 4(b). In contrast, the sequence $(\mathit{Insert}, \mathit{Delete})$ (Figure 4(a)) is allowed. The net effects of these edit operation sequences are identical, in that they both skip one word each from the two sequences to be aligned. As a result, there is no use in discriminating between these two, and one of them, namely $(\mathit{Delete}, \mathit{Insert})$, is prohibited.

[1] For a vertex v at the border of an edit graph, some edit operations are not applicable (e.g., *Insert* and *Substitute* at vertices on the right border in Figure 2); we say such operations are *inadmissible* at v. Otherwise, an edit operation is *admissible*.

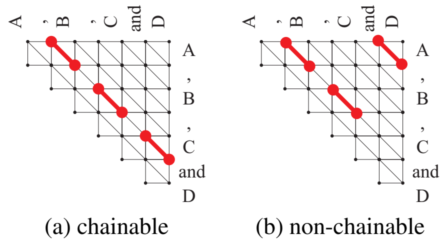

Figure 5: A coordination with four conjuncts represented as (a) chainable, and (b) non-chainable partial paths. We take (a) as the canonical representation.

3.2 Learning task

By the restriction of condition (iii) introduced above and the omission of (*Outside*, *Substitute*) from the node types, we can uniquely determine the complete path (from the initial node to the terminal node) that conjoins all the local alignments by *Outside* nodes (which corresponds to edges in the original edit graph). In Figure 2, the augmented *Outside* edges in this unique path are plotted as dotted lines for illustration.

Thus we obtain a complete path which is compatible with Collins's perceptron-based sequence learning method. The objective of the learning algorithms, which we will describe in Section 4, is to optimize the weight of features so that running the Viterbi algorithm will yield the same path as the gold standard.

Because a node in our state space corresponds to an edge in the original edit graph (see Figure 3), an arc in our state space is actually a pair of consecutive edges (or equivalently, edit operations) in the original graph. Hence our model is more expressive than the original edit graph in that the score function can have a term (feature) defined on a pair of edit operations instead of one.

3.3 More complex coordinations

Even if a coordination comprises three or more conjuncts, our model can handle them, as it can be represented as a set of pairwise local alignments that are *chainable* (Gusfield, 1997, Section 13.3). If pairwise local alignments are chainable, a unique complete path that conjoins all these alignments can be determined, allowing the same treatment as the case with two conjuncts.

For instance, a coordination with four conjuncts (A, B, C and D) can be decomposed into a set of pairwise alignments $\{(A,B),(B,C),(C,D)\}$ as depicted in Figure 5(a). This set of alignments are chainable and thus constitute the canonical encoding for this coordination; any other pairwise decomposition for these four conjuncts, like $\{(A,B),(B,C),(A,D)\}$ (Figure 5(b)), is not chainable.

Our model can handle multiple non-nested coordinations in a single sentence as well, as they can also be decomposed into chainable pairwise alignments. It cannot encode nested coordinations like (A, B, and (C and D)), however.

4 Algorithms

4.1 Reducing the cost of training data construction

Our learning method is supervised, meaning that it requires training data annotated with correct labels. Since a label in our problem is local alignments (or paths in an edit graph) representing coordinations, the training sentences have to be annotated with word-by-word alignments.

There are two reasons relaxing this requirement is desirable. First, it is expensive to construct such data. Second, there are coordinate conjunctions in which word-by-word correspondence is unclear even for humans. In Figure 2, for example, a word-by-word alignment of 'standard' with 'dense' is depicted, but it might be more natural to regard a word 'standard' as being aligned with two words 'dose dense' combined together.

Even if word-by-word alignment is uncertain, the boundaries of conjuncts are often obvious, and it is also much easier for human annotators to mark only the beginning and end of each conjunct. Thus we would like to allow for training examples in which only alignment boundaries are specified, instead of a full word-by-word alignment.

For these examples, conjunct boundaries corresponds to a rectangular region rather than a single path in an edit graph. The shaded box in Figure 2 illustrates the rectangular region determined by the boundaries of an alignment between the phrases "182% for the dose dense arm" and "99% for the standard arm." There are many possible alignment paths in this box, among which we do not know which one is correct (or even likely). To deal with

input: Set of examples $S = \{(x_i, Y_i)\}$
Iteration cutoff T
output: Averaged weight vector $\bar{w}$

1: $\bar{w} \leftarrow 0$; $w \leftarrow 0$
2: **for** $t \leftarrow 1 \dots T$ **do**
3: $\quad \Delta w \leftarrow 0$
4: $\quad$ **for each** $(x_i, Y_i) \in S$ **do**
5: $\quad\quad y \leftarrow \arg\max_{y \in Y_i} w \cdot f(x_i, y)$
6: $\quad\quad y' \leftarrow \arg\max_{y \in A(x_i)} w \cdot f(x_i, y)$
7: $\quad\quad \Delta f \leftarrow f(x_i, y) - f(x_i, y')$
8: $\quad\quad \Delta w \leftarrow \Delta w + \Delta f$
9: $\quad$ **end for**
10: $\quad$ **if** $\Delta w = 0$ **then**
11: $\quad\quad$ **return** $\bar{w}$
12: $\quad$ **end if**
13: $\quad w \leftarrow w + \Delta w$
14: $\quad \bar{w} \leftarrow [(t-1)\bar{w} + w]/t$
15: **end for**
16: **return** $\bar{w}$

Figure 6: Path-based algorithm

this difficulty, we propose two simple heuristics we call the (i) path-based and (ii) box-based methods. As mentioned earlier, both of these methods are based on Collins's averaged-perceptron algorithm for sequence labeling (Collins, 2002).

4.2 Path-based method

Our first method, which we call the "path-based" algorithm, is shown in Figure 6. We denote by $A(x)$ all possible alignments (paths) over x. The algorithm receives T, the maximum number of iterations, and a set of examples $S = \{(x_i, Y_i)\}$ as input, where x_i is a sentence (a sequence of words with their attributes, e.g., part-of-speech, lemma, prefixes, and suffixes) and $Y_i \subset A(x_i)$ is the set of admissible alignments (paths) for x_i. When a sentence is fully annotated with a word-by-word alignment y, $Y_i = \{y\}$ is a singleton set. In general boundary-only examples we described in Section 4.1, Y_i holds all possible alignments compatible with the marked range, or equivalently, paths that pass through the upper-left and lower-right corners of a rectangular region. Note that it is not necessary to explicitly enumerate all the member paths of Y_i; the set notation here is only for the sake of presentation.

The external function $f(x, y)$ returns a vector (called the *global feature vector* in (Sha and Pereira, 2003)) of the number of feature occurrences along the alignment path y. In the beginning (line 5 in the figure) of the inner loop, the target path (alignment)

input: Set of examples $S = \{(x_i, Y_i)\}$
Iteration cutoff T
output: Averaged weight vector $\bar{w}$

1: $\bar{w} \leftarrow 0$; $w \leftarrow 0$
2: **for each** $(x_i, Y_i) \in S$ **do**
3: $\quad g_i \leftarrow (1/|Y_i|) \sum_{y \in Y_i} f(x_i, y)$
4: **end for**
5: **for** $t \leftarrow 1 \dots T$ **do**
6: $\quad \Delta w \leftarrow 0$
7: $\quad$ **for each** $(x_i, Y_i) \in S$ **do**
8: $\quad\quad y' \leftarrow \arg\max_{y \in A(x_i)} w \cdot f(x_i, y)$
9: $\quad\quad$ Convert y' into its box representation Y'
10: $\quad\quad g' \leftarrow (1/|Y_i'|) \sum_{y \in Y_i'} f(x_i, y)$
11: $\quad\quad \Delta f \leftarrow g_i - g'$
12: $\quad\quad \Delta w \leftarrow \Delta w + \Delta f$
13: $\quad$ **end for**
14: $\quad$ **if** $\Delta w = 0$ **then**
15: $\quad\quad$ **return** $\bar{w}$
16: $\quad$ **end if**
17: $\quad w \leftarrow w + \Delta w$
18: $\quad \bar{w} \leftarrow [(t-1)\bar{w} + w]/t$
19: **end for**
20: **return** $\bar{w}$

Figure 7: Box-based algorithm

is recomputed with the current weight vector w. The arg max in lines 5 and 6 can be computed efficiently ($O(n^2)$, where n is the number of words in x) by running a pass of the Viterbi algorithm in the edit graph for x. The weight vector w varies between iterations, and so does the most likely alignment with respect to w. Hence the recomputation in line 5 is needed.

4.3 Box-based method

Our next method, called "box-based," is designed on the following heuristic. Given a rectangle region representing a local alignment (hence all nodes in the region are of polarity *Inside*) in an edit graph, we distribute feature weights in proportion to the probability of a node (or an arc) being passed by a path from the initial (upper left) node to the terminal (lower right) node of the rectangle. We assume paths are uniformly distributed.

Figure 8 displays an 8×8 sub-grid of an edit graph. The figure under each vertex shows the number of paths passing through the vertex. Vertices near the upper-left and the lower-right corner have a large frequency, and the frequency drops exponentially towards the top right corner and the bottom left corner, hence placing a strong bias on the paths near diagonals. This distribution fits our preference

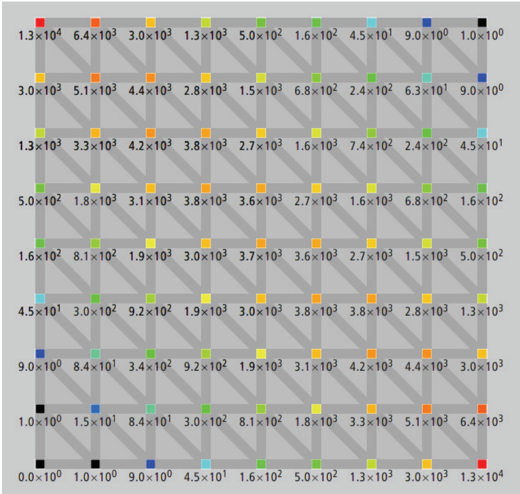

Figure 8: Number of paths passing through the vertices of an 8×8 grid.

towards alignments with a larger number of substitutions.

The pseudo-code for the box-based algorithm is shown in Figure 7. For each example x_i and its possible target labels (alignments) Y_i, this algorithm first (line 3) computes and stores in the vector g_i the average number of feature occurrences in all possible target paths in Y_i. This quantity can be computed simply by summing over all nodes and edges feature occurrences multiplied by the pre-computed frequency of each nodes and arcs at which these features occur. analogously to the forward-backward algorithm. In each iteration, the algorithm scans every example (lines 7–13), computing the Viterbi path y' (line 8) according to the current weight vector w. Line 9 then converts y' to its box representation Y', by sequentially collapsing consecutive *Inside* nodes in y' as a box. For instance, let y' be the local alignment depicted as the bold line in Figure 2. The box Y' computed in line 9 for this y' is the shaded area in the figure. In parallel to the initialization step in line 3, we store in g' the average feature occurrences in Y' and update the current weight vector w by the difference between the target g_i and g'. These steps can be interpreted as a Viterbi approximation for computing the optimal set Y' of alignments directly.

5 Related work

5.1 Discriminative learning of edit distance

In our model, the state space of sequence alignment, or edit graph, is two-dimensional (which is actually three-dimensional if the dimension for labels is taken into account). This is contrastive to the one dimensional models used by Collins's perceptron-based sequence method (Collins, 2002) which our algorithms are based upon, and by the linear-chain CRFs.

McCallum et al. (McCallum et al., 2005) proposed a CRF tailored to learning string edit distance for the identity uncertainty problem. The state space in their work is two dimensional just like our model, but it is composed of two decoupled subspaces, each corresponding to 'match' and 'mismatch,' thus sharing only the initial state. It is not possible to make a transition from a state in the 'match' state space to the 'mismatch' space (and vice versa). As we can see from the decoupled state space, this method is based on global alignment rather than local alignment; it is not clear whether their method can identify local homologies in sequences. Our method uses a single state space in which both 'match (inside)' and 'mismatch (outside)' nodes co-exist and transition between them is permitted.

5.2 Inverse sequence alignment in computational biology

In computational biology, the estimation of a substitution matrix from data is called the *inverse sequence alignment* problem. Until recently, there have been a relatively small number of papers in this field despite a large body of literature in sequence alignment. Theoretical studies in the inverse sequence alignment include (Pachter and Sturmfels, 2004; Sun et al., 2004). Recently, CRFs have been applied for optimizing the substitution matrix in the context of global protein sequence alignment (Do et al., 2006).

6 Empirical evaluation

6.1 Dataset and Task

We used the GENIA Treebank beta corpus (Kim et al., 2003)[2] for evaluation of our methods. The cor-

[2]http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA

pus consists of 500 parsed abstracts in Medline with a total of 4529 sentences.

Although the Penn Treebank Wall Street Journal (WSJ) is the de facto standard corpus for evaluating chunking and parsing performance, it lacks adequate structural information on coordinate conjunctions, and therefore does not serve our purpose. Many coordinations in the Penn Treebank are given a flat bracketing like (A, B, and C D), and thus we cannot tell which of ((A, B, and C) D) and ((A), (B), and (C D)) gives a correct alignment. The GENIA corpus, in contrast, distinguishes ((A, B, and C) D) and ((A), (B), and (C D)) explicitly, by providing more detailed bracketing. In addition, the corpus contains an explicit tag "COOD" for marking coordinations.

To avoid nested coordinations, which admittedly require techniques other than the one proposed in this paper, we selected from the GENIA corpus sentences in which the conjunction "and" occurs just once. After this operation, the number of sentences reduced to 1668, from which we further removed 32 that are not associated with the 'COOD' tag, and 3 more whose annotated tree structures contained obvious errors. Of the remaining 1633 sentences, 1061 were coordinated noun phrases annotated with NP-COOD tags, 226 coordinated verb phrases (VP-COOD), 142 coordinated adjective phrases (ADJP-COOD), and so on. Because the number of VP-COOD, ADJP-COOD, and other types of coordinated phrases are too small to make a meaningful benchmark, we focus on coordinated noun phrases in this experiment.

The task hence amounts to identifying coordinated NPs and their constituent conjuncts in the 1633 sentences, all of which contain a coordination marker "and" but only 1061 of which are actually coordinated NPs.

6.2 Baselines

We used several publicly available full parsers as baselines: (i) the Bikel parser (Bikel, 2005) version 0.9.9c with configuration file `bikel.properties` (denoted as Bikel/Bikel), (ii) the Bikel parser in the Collins parser emulation mode (using `collins.properties` file) (Bikel/Collins), and (iii) Charniak and Johnson's reranking parser (Charniak-Johnson) (Charniak and Johnson, 2005). We trained Bikel's parser and its Collins emulator with the GENIA corpus, WSJ, and the combination of the two. Charniak and Johnson's parser was used as distributed at Charniak's home page (and is WSJ trained).

Another baseline we used is chunkers based on linear-chain CRFs and the standard BIO labels. We trained two types of CRF-based chunkers by using different BIO sequences, one for the conjunct bracketing and the other for coordination bracketing. The chunkers were implemented with T. Kudo's CRF++ package version 0.45. We varied its regularization parameters C among $C \in \{0.01, 0.1, 1, 10, 100, 1000\}$, and the best results among these are reported below.

6.3 Features

Let $x = (x_1, \ldots, x_n)$ be a sentence, with its member x_k a vector of attributes for the kth word. The attributes include word surface, part-of-speech (POS), and suffixes, among others.

Table 1 summarizes (i) the features assigned to a node whose corresponding edge in the original edit graph for x is emanating from row i and column j, and (ii) the features assigned to the arcs (consisting of two edges in the original edit graph) whose joint (the vertex between the two edges) is a vertex at row i and column j.

We also tested the path-based and box-based methods, and the CRF chunkers both with and without the word and suffix features.

Although this is not a requirement of our model or algorithms, every feature we use in this experiment is binary; if the condition associated with a feature is satisfied, the feature takes a value of 1; otherwise, it is 0. A condition typically asks whether or not specific attributes match those at a current node, arc, or their neighbors.

We used the POS tags from the GENIA corpus as the POS attribute. The morphological features include 3- and 4-gram suffixes and indicators of whether a word includes capital letters, hyphens, and digits.

For the baseline CRF-based chunkers, we assign the word, POS (from GENIA), and the morphological features to nodes, and the POS features to edges. The feature set is identical to those used for our proposed methods, except for features defined on row-column combination (i.e., those defined over both i

Table 1: Features for the proposed methods

***Substitute* (diagonal) nodes** $(*, \mathit{Substitute}, *)$	Indicators of the word, POS, and morphological attributes of x_i, x_j, (x_{i-1}, x_i), (x_i, x_{i+1}), (x_{j-1}, x_j), (x_j, x_{j+1}), and (x_i, x_j), respectively combined with the type of the node.
	For each of the word, POS, and morphological attributes, an indicator of whether the respective attribute is identical in x_i and x_j, combined with the type of the node.
***Delete* (vertical) nodes** $(*, \mathit{Delete}, *)$	Indicators of the word, POS, and morphological attributes of x_i, x_j, x_{j-1}, (x_{i-1}, x_i), (x_i, x_{i+1}), and (x_{j-1}, x_j), combined with the type of the node.
***Insert* (horizontal) nodes** $(*, \mathit{Insert}, *)$	Indicators of the word, POS, and morphological attributes of x_i, x_{i-1}, x_j, (x_{i-1}, x_i), (x_{j-1}, x_j), and (x_j, x_{j+1}), combined with the type of the node.
Any arcs $(*,*,*) \rightarrow (*,*,*)$	Indicators of the POS attribute of x_i, x_{i-1}, x_j, x_{j-1}, (x_{i-2}, x_{i-1}), (x_{i-1}, x_i), (x_i, x_{i+1}), (x_{j-2}, x_{j-1}), (x_{j-1}, x_j), (x_j, x_{j+1}), (x_{i-1}, x_{j-1}), (x_{i-1}, x_j), (x_i, x_{j-1}) and (x_i, x_j), combined with the type pair of the arc.
Arcs between nodes of different polarity $(*, \mathit{Inside}, *) \rightarrow (*, \mathit{Outside}, *)$ and $(*, \mathit{Outside}, *) \rightarrow (*, \mathit{Inside}, *)$	Indicator of the distance $j - i$ between two words x_i and x_j, combined with the type pair of the arc.

and j in Table 1. The latter cannot be incorporated as a local features in chunkers based on linear chain.

For the Bikel (and its Collins emulation) parsers which accepts POS tags output by external taggers upon testing, we gave them the POS tags from the GENIA corpus, for fair comparison with the proposed methods and CRF-based chunkers.

6.4 Evaluation criteria

We employed two evaluation criteria: (i) correctness of the conjuncts output by the algorithm, and (ii) correctness of the range of coordinations as a whole.

For the correctness of conjuncts, we further use two evaluation criteria. The first evaluation method ("pairwise evaluation") is based on the decomposition of coordinations into the canonical set of pairwise alignments, as described in Section 3.3. After the set of pairwise alignments is obtained, each pairwise alignment is transformed into a box surrounded by their boundaries. Using these boxes, we evaluate precision, recall and F rates through the following definition. The precision measures how many of the boxes output by the algorithm exactly match those in the gold standard, and the recall rate is the percentage of boxes found by the algorithm. The F rate is the harmonic mean of the precision and the recall.

The second evaluation method ("chunk-based evaluation") for conjuncts is based on whether the algorithm correctly outputs the beginning and end of each conjunct, in the same manner as the chunking tasks. Here, we adopt the evaluation criteria for the CoNLL 99 NP bracketing task[3]; the precision equals how many of the NP conjuncts output by the algorithm are correct, and the recall is the percentage of NP conjuncts found by the algorithm.

Of these two evaluation methods for conjuncts, it is harder to obtain a higher pairwise evaluation score than the chunk-based evaluation. To be counted as a true positive in the pairwise evaluation, two consecutive chunks must be output correctly by the algorithm.

For the correctness of the coordination range, we check if both the start of the first coordinated conjunct and the end of the last conjunct in the gold match those output by the algorithm The reason we evaluate coordination range is to compare our proposed method with the full parsers trained on WSJ (but applied to GENIA). Although WSJ and GENIA differ in the way conjuncts are annotated, they are mostly identical on how the range of coordinations are annotated, and hence comparison is feasible in terms of coordination range. For the baseline parsers, we regard the bracketing directly surrounding the coordination marker "and" as their output.

In (Clegg and Shepherd, 2007), an F score of 75.5 is reported for the Bikel parser on coordination detection. Their evaluation is based on dependencies, which is different from our evaluation criteria which are all based on boundaries. Generally speaking, our evaluation criterion seems stricter, as exemplified in Figures 7 and 8 of Clegg and Shepherd's paper; in these figures, our evaluation criterion would result

[3] http://www.cnts.ua.ac.be/conll99/npb/

Table 2: Performance on conjunct bracketing. P: precision (%), R: recall (%), F: F rate.

	Pairwise evaluation			Chunk-based evaluation		
Method	P	R	F	P	R	F
Path-based method	61.4	56.2	58.7	70.9	66.9	68.9
Path-based method without word and suffix features	61.7	**58.8**	**60.2**	**71.2**	**69.7**	**70.5**
Box-based method	60.6	58.3	59.4	70.5	69.1	69.8
Box-based method without word and suffix features	59.5	58.3	58.9	69.7	69.5	69.6
Linear-chain CRF chunker (conjunct bracketing)	**62.6**	51.4	56.4	71.0	66.1	68.5
Bikel/Collins, trained with GENIA	50.0	48.6	49.3	65.0	64.2	64.6
Bikel/Bikel, trained with GENIA	50.1	47.8	49.0	63.9	61.3	62.6

Table 3: Performance on coordination bracketing. P: precision (%), R: recall (%), F: F rate.

Method	P	R	F
Path-based method	58.2	55.3	56.7
Path-based method without words and suffix features	57.7	**56.6**	**57.2**
Box-based method	55.6	54.4	55.0
Box-based method without words and suffix features	54.8	54.6	54.7
Linear-chain CRF chunker, trained with conjunct bracketing	43.9	46.7	45.3
Linear-chain CRF chunker, trained with coordination bracketing	**58.4**	51.0	54.5
Bikel/Collins, trained with GENIA	44.0	45.4	44.7
Bikel/Collins, trained with WSJ	42.3	43.2	42.7
Bikel/Collins, trained with GENIA+WSJ	43.3	45.1	44.1
Bikel/Bikel, trained with GENIA	44.8	45.4	45.1
Bikel/Bikel, trained with WSJ	40.7	41.5	41.1
Bikel/Bikel, trained with GENIA+WSJ	43.9	45.8	44.9
Charniak-Johnson reranking parser	48.3	45.2	46.7

in zero true positive, whereas their evaluation counts the dependency arc from ‘genes’ to ‘human’ as one true positive.

6.5 Results

The results of conjunct and coordination bracketing are shown in Tables 2 and 3, respectively. These are the results of a five-fold cross validation. We ran the proposed methods until convergence or the cutoff iteration of $T = 10000$, whichever comes first.

The path-based method (without words and suffixes) and box-based method (with full features) each achieved 2.0 and 1.3 point improvements over the CRF chunker in terms of the F score in conjunct identification (chunk-based evaluation), 3.8 and 3.0 point improvement in terms of pairwise evaluation, and 2.7 and 0.5 points in coordinate identification, respectively. Our methods also showed a performance considerably higher than the baseline parsers.

The performance of the path-based method was better when the word and suffix features were removed, while the box-based method and CRF chunkers performed better with these features.

7 Conclusions

We have proposed a new coordination learning and disambiguation method that can incorporate many different features, and automatically optimize their weights on training data.

In the experiment of Section 6, the proposed method obtained a performance superior to a linear-chain chunker and to the state-of-art full parsers.

We used only syntactic and morphological features, and did not use external similarity measures like thesauri and corpora, although they are reported to be effective for disambiguating coordinations. We note that it is easy to incorporate such external similarity measures as a feature in our model, thanks to its two-dimensional state space. The similarity of two words derived from an external knowledge base can be assigned to a *Substitute* node at a corresponding location in the state space in a straightforward manner. This is a topic we are currently working on.

We are also planning to reimplement our algorithms using CRFs instead of the averaged perceptron algorithm.

References

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

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

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine *n*-best parsing and MaxEnt discriminative reranking. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL-2005)*.

Andrew B Clegg and Adrian J Shepherd. 2007. Benchmarking natural-language parsers for biological applications using dependency graphs. *BMC Bioinformatics*, 8(24).

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)*.

C. B. Do, S. S. Gross, and S. Batzoglou. 2006. CONTRAlign: discriminative training for protein sequence alignment. In *Proceedings of the Tenth Annual International Conference on Computational Molecular Biology (RECOMB 2006)*.

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

J.-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. 2003. GENIA corpus: a semantically annotated corpus for biotextmining. *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.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML-2001)*, pages 282–289. Morgan Kaufmann.

Andrew McCallum, Kedar Bellare, and Fernando Pereira. 2005. A conditional random field for discriminatively-trained finite-state string edit distance. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI-2005)*.

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

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.

Lior Pachter and Bernd Sturmfels. 2004. Parametric inference for biological sequence analysis. *Proceedings of the National Academy of Sciences of the USA*, 101(46):16138–16143.

Philip Resnik. 1999. Semantic similarity in a taxonomy: an information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the Human Language Technology Conference North American Chapter of Association for Computational Linguistics (HLT-NAACL 2003)*, pages 213–220, Edmonton, Alberta, Canada. Association for Computational Linguistics.

Fangting Sun, David Fernández-Baca, and Wei Yu. 2004. Inverse parametric sequence alignment. *Journal of Algorithms*, 53:36–54.