A Linear-Time Bottom-Up Discourse Parser
with Constraints and Post-Editing

Vanessa Wei Feng
Department of Computer Science
University of Toronto
Toronto, ON, Canada
weifeng@cs.toronto.edu

Graeme Hirst
Department of Computer Science
University of Toronto
Toronto, ON, Canada
gh@cs.toronto.edu

Abstract

Text-level discourse parsing remains a challenge. The current state-of-the-art overall accuracy in relation assignment is 55.73%, achieved by Joty et al. (2013). However, their model has a high order of time complexity, and thus cannot be applied in practice. In this work, we develop a much faster model whose time complexity is linear in the number of sentences. Our model adopts a greedy bottom-up approach, with two linear-chain CRFs applied in cascade as local classifiers. To enhance the accuracy of the pipeline, we add additional constraints in the Viterbi decoding of the first CRF. In addition to efficiency, our parser also significantly outperforms the state of the art. Moreover, our novel approach of post-editing, which modifies a fully-built tree by considering information from constituents on upper levels, can further improve the accuracy.

1 Introduction

Discourse parsing is the task of identifying the presence and the type of the discourse relations between discourse units. While research in discourse parsing can be partitioned into several directions according to different theories and frameworks, Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) is probably the most ambitious one, because it aims to identify not only the discourse relations in a small local context, but also the hierarchical tree structure for the **full text**: from the relations relating the smallest discourse units (called elementary discourse units, EDUs), to the ones connecting paragraphs.

For example, Figure 1 shows a text fragment consisting of two sentences with four EDUs in total (e_1-e_4). Its discourse tree representation is

shown below the text, following the notation convention of RST: the two EDUs e_1 and e_2 are related by a mononuclear relation CONSEQUENCE, where e_2 is the more salient span (called *nucleus*, and e_1 is called *satellite*); e_3 and e_4 are related by another mononuclear relation CIRCUMSTANCE, with e_4 as the nucleus; the two spans $e_{1:2}$ and $e_{3:4}$ are further related by a multi-nuclear relation SEQUENCE, with both spans as the nucleus.

Conventionally, there are two major sub-tasks related to text-level discourse parsing: (1) **EDU segmentation**: to segment the raw text into EDUs, and (2) **tree-building**: to build a discourse tree from EDUs, representing the discourse relations in the text. Since the first sub-task is considered relatively easy, with the state-of-art accuracy at above 90% (Joty et al., 2012), the recent research focus is on the second sub-task, and often uses manual EDU segmentation.

The current state-of-the-art overall accuracy of the tree-building sub-task, evaluated on the RST Discourse Treebank (RST-DT, to be introduced in Section 8), is 55.73% by Joty et al. (2013). However, as an optimal discourse parser, Joty et al.'s model is highly inefficient in practice, with respect to both their DCRF-based local classifiers, and their CKY-like bottom-up parsing algorithm. DCRF (Dynamic Conditional Random Fields) is a generalization of linear-chain CRFs, in which each time slice contains a set of state variables and edges (Sutton et al., 2007). CKY parsing is a bottom-up parsing algorithm which searches all possible parsing paths by dynamic programming. Therefore, despite its superior performance, their model is infeasible in most realistic situations.

The main objective of this work is to develop a more efficient discourse parser, with similar or even better performance with respect to Joty et al.'s optimal parser, but able to produce parsing results in real time.

Our contribution is three-fold. First, with a

511

[**On Aug. 1, the state tore up its controls,**]e_1 [**and food prices leaped**]e_2 [**Without buffer stocks,**]e_3 [**inflation exploded.**]e_4
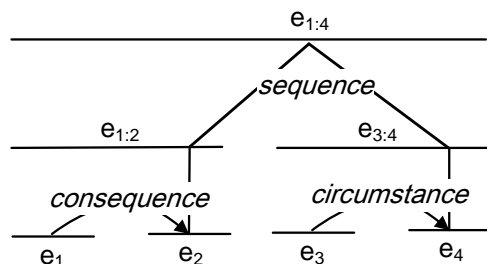
Figure 1: An example text fragment composed of two sentences and four EDUs, with its RST discourse tree representation shown below.

greedy bottom-up strategy, we develop a discourse parser with a time complexity linear in the total number of sentences in the document. As a result of successfully avoiding the expensive non-greedy parsing algorithms, our discourse parser is very efficient in practice. Second, by using two linear-chain CRFs to label a sequence of discourse constituents, we can incorporate contextual information in a more natural way, compared to using traditional discriminative classifiers, such as SVMs. Specifically, in the Viterbi decoding of the first CRF, we include additional constraints elicited from common sense, to make more effective local decisions. Third, after a discourse (sub)tree is fully built from bottom up, we perform a novel post-editing process by considering information from the constituents on upper levels. We show that this post-editing can further improve the overall parsing performance.

2   Related work

2.1   HILDA discourse parser

The HILDA discourse parser by Hernault et al. (2010) is the first attempt at RST-style text-level discourse parsing. It adopts a pipeline framework, and greedily builds the discourse tree from the bottom up. In particular, starting from EDUs, at each step of the tree-building, a binary SVM classifier is first applied to determine which pair of adjacent discourse constituents should be merged to form a larger span, and another multi-class SVM classifier is then applied to assign the type of discourse relation that holds between the chosen pair.

The strength of HILDA's greedy tree-building strategy is its efficiency in practice. Also, the employment of SVM classifiers allows the incorporation of rich features for better data representation (Feng and Hirst, 2012). However, HILDA's approach also has obvious weakness: the greedy algorithm may lead to poor performance due to local optima, and more importantly, the SVM classifiers are not well-suited for solving structural problems due to the difficulty of taking context into account.

2.2   Joty et al.'s joint model

Joty et al. (2013) approach the problem of text-level discourse parsing using a model trained by Conditional Random Fields (CRF). Their model has two distinct features.

First, they decomposed the problem of text-level discourse parsing into two stages: intra-sentential parsing to produce a discourse tree for each sentence, followed by multi-sentential parsing to combine the sentence-level discourse trees and produce the text-level discourse tree. Specifically, they employed two separate models for intra- and multi-sentential parsing. Their choice of two-stage parsing is well motivated for two reasons: (1) it has been shown that sentence boundaries correlate very well with discourse boundaries, and (2) the scalability issue of their CRF-based models can be overcome by this decomposition.

Second, they jointly modeled the structure and the relation for a given pair of discourse units. For example, Figure 2 shows their intra-sentential model, in which they use the bottom layer to represent discourse units; the middle layer of binary nodes to predict the connection of adjacent discourse units; and the top layer of multi-class nodes to predict the type of the relation between two units. Their model assigns a probability to each possible constituent, and a CKY-like parsing algorithm finds the globally optimal discourse tree, given the computed probabilities.

The strength of Joty et al.'s model is their joint modeling of the structure and the relation, such that information from each aspect can interact with the other. However, their model has a major defect in its inefficiency, or even infeasibility, for application in practice. The inefficiency lies in both their DCRF-based joint model, on which inference is usually slow, and their CKY-like parsing algorithm, whose issue is more prominent. Due to the $O(n^3)$ time complexity, where n is the number
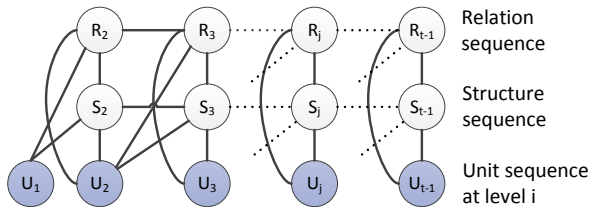
Figure 2: Joty et al. (2013)'s intra-sentential Condition Random Fields.

of input discourse units, for large documents, the parsing simply takes too long[1].

3 Overall work flow

Figure 3 demonstrates the overall work flow of our discourse parser. The general idea is that, similar to Joty et al. (2013), we perform a sentence-level parsing for each sentence first, followed by a text-level parsing to generate a full discourse tree for the whole document. However, in addition to efficiency (to be shown in Section 6), our discourse parser has a distinct feature, which is the post-editing component (to be introduced in Section 5), as outlined in dashes.

Our discourse parser works as follows. A document D is first segmented into a list of sentences. Each sentence S_i, after being segmented into EDUs (not shown in the figure), goes through an intra-sentential bottom-up tree-building model M_{intra}, to form a sentence-level discourse tree T_{S_i}, with the EDUs as leaf nodes. After that, we apply the intra-sentential post-editing model P_{intra} to modify the generated tree T_{S_i} to $T_{S_i}^p$, by considering upper-level information.

We then combine all sentence-level discourse tree $T_{S_i}^p$'s using our multi-sentential bottom-up tree-building model M_{multi} to generate the text-level discourse tree T_D. Similar to sentence-level parsing, we also post-edit T_D using P_{multi} to produce the final discourse tree T_D^p.

---

[1] The largest document in the RST-DT contains over 180 sentences, i.e., $n > 180$ for their multi-sentential CKY parsing. Intuitively, suppose the average time to compute the probability of each constituent is 0.01 second, then in total, the CKY-like parsing takes over 16 hours. It is possible to optimize Joty et al.'s CKY-like parsing by replacing their CRF-based computation for upper-level constituents with some local computation based on the probabilities of lower-level constituents. However, such optimization is beyond the scope of this paper.

4 Bottom-up tree-building

For both intra- and multi-sentential parsing, our bottom-up tree-building process adopts a similar greedy pipeline framework like the HILDA discourse parser (discussed in Section 2.1), to guarantee efficiency for large documents. In particular, starting from the constituents on the bottom level (EDUs for intra-sentential parsing and sentence-level discourse trees for multi-sentential parsing), at each step of the tree-building, we greedily merge a pair of adjacent discourse constituents such that the merged constituent has the highest probability as predicted by our *structure* model. The *relation* model is then applied to assign the relation to the new constituent.

4.1 Linear-chain CRFs as Local models

Now we describe the local models we use to make decisions for a given pair of adjacent discourse constituents in the bottom-up tree-building. There are two dimensions for our local models: (1) scope of the model: intra- or multi-sentential, and (2) purpose of the model: for determining structures or relations. So we have four local models, M_{intra}^{struct}, M_{intra}^{rel}, M_{multi}^{struct}, and M_{multi}^{rel}.

While our bottom-up tree-building shares the greedy framework with HILDA, unlike HILDA, our local models are implemented using CRFs. In this way, we are able to take into account the sequential information from contextual discourse constituents, which cannot be naturally represented in HILDA with SVMs as local classifiers. Therefore, our model incorporates the strengths of both HILDA and Joty et al.'s model, i.e., the efficiency of a greedy parsing algorithm, and the ability to incorporate sequential information with CRFs.

As shown by Feng and Hirst (2012), for a pair of discourse constituents of interest, the sequential information from contextual constituents is crucial for determining structures. Therefore, it is well motivated to use Conditional Random Fields (CRFs) (Lafferty et al., 2001), which is a discriminative probabilistic graphical model, to make predictions for a sequence of constituents surrounding the pair of interest.

In this sense, our local models appear similar to Joty et al.'s non-greedy parsing models. However, the major distinction between our models and theirs is that we do not jointly model the structure and the relation; rather, we use two linear-
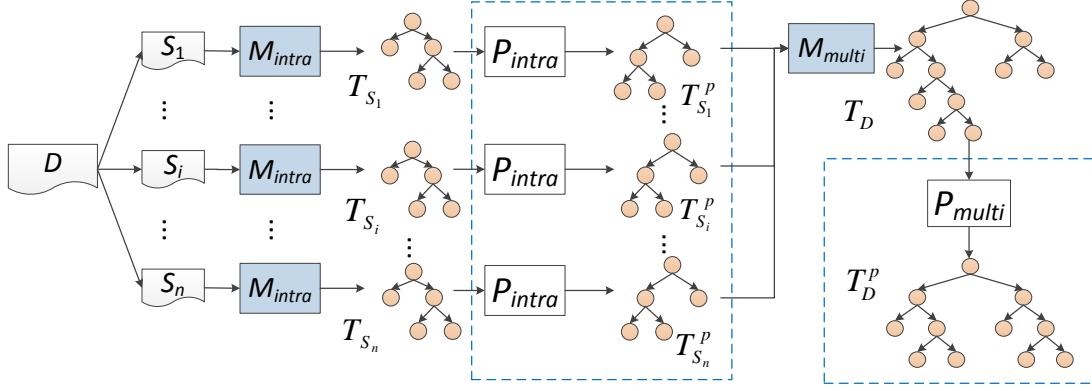
Figure 3: The work flow of our proposed discourse parser. In the figure, M_{intra} and M_{multi} stand for the intra- and multi-sentential bottom-up tree-building models, and P_{intra} and P_{multi} stand for the intra- and multi-sentential post-editing models.

chain CRFs to model the structure and the relation separately. Although joint modeling has shown to be effective in various NLP and computer vision applications (Sutton et al., 2007; Yang et al., 2009; Wojek and Schiele, 2008), our choice of using two separate models is for the following reasons:

First, it is not entirely appropriate to model the structure and the relation at the same time. For example, with respect to Figure 2, it is unclear how the relation node R_j is represented for a training instance whose structure node $S_j = 0$, i.e., the units U_{j-1} and U_j are disjoint. Assume a special relation NO-REL is assigned for R_j. Then, in the tree-building process, we will have to deal with the situations where the joint model yields conflicting predictions: it is possible that the model predicts $S_j = 1$ and $R_j = $ NO-REL, or vice versa, and we will have to decide which node to trust (and thus in some sense, the structure and the relation is no longer jointly modeled).

Secondly, as a joint model, it is mandatory to use a dynamic CRF, for which exact inference is usually intractable or slow. In contrast, for linear-chain CRFs, efficient algorithms and implementations for exact inference exist.

4.2 Structure models

4.2.1 Intra-sentential structure model

Figure 4a shows our intra-sentential structure model M_{intra}^{struct} in the form of a linear-chain CRF. Similar to Joty et al.'s intra-sentential model, the first layer of the chain is composed of discourse constituents U_j's, and the second layer is composed of binary nodes S_j's to indicate the probability of merging adjacent discourse constituents.

(a) Intra-sentential structure model M_{intra}^{struct}.

(b) Multi-sentential structure model M_{multi}^{struct}. C^1, C^2, and C^3 denote the three chains for predicting U_j and U_{j+1}.
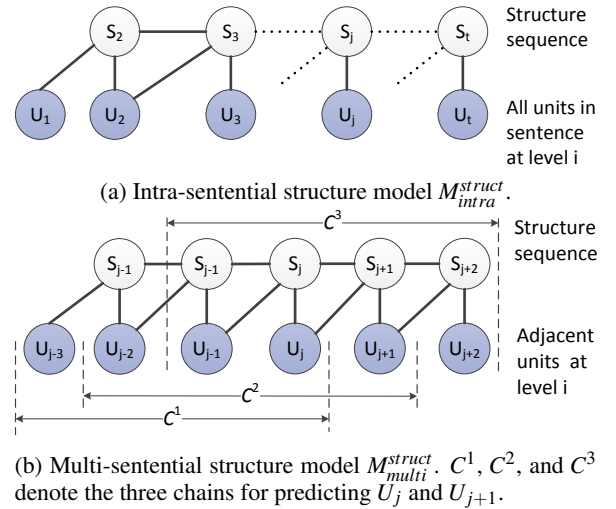
Figure 4: Local structure models.

At each step in the bottom-up tree-building process, we generate a single sequence E, consisting of $U_1, U_2, \ldots, U_j, \ldots, U_t$, which are all the current discourse constituents in the sentence that need to be processed. For instance, initially, we have the sequence $E_1 = \{e_1, e_2, \ldots, e_m\}$, which are the EDUs of the sentence; after merging e_1 and e_2 on the second level, we have $E_2 = \{e_{1:2}, e_3, \ldots, e_m\}$; after merging e_4 and e_5 on the third level, we have $E_3 = \{e_{1:2}, e_3, e_{4:5}, \ldots, e_m\}$, and so on.

Because the structure model is the first component in our pipeline of local models, its accuracy is crucial. Therefore, to improve its accuracy, we enforce additional commonsense constraints in its Viterbi decoding. In particular, we disallow 1-1 transitions between adjacent labels (a discourse unit can be merged with at most one adjacent unit), and we disallow all-zero sequences (at least one

pair must be merged).

Since the computation of E_i **does not** depend on a particular pair of constituents, we can use the same sequence E_i to compute structural probabilities for **all** adjacent constituents. In contrast, Joty et al.'s computation of intra-sentential sequences depends on the particular pair of constituents: the sequence is composed of the pair in question, with other EDUs in the sentence, even if those EDUs have already been merged. Thus, different CRF chains have to be formed for different pairs of constituents. In addition to efficiency, our use of a single CRF chain for all constituents can better capture the sequential dependencies among context, by taking into account the information from partially built discourse constituents, rather than bottom-level EDUs only.

4.2.2 Multi-sentential structure model

For multi-sentential parsing, where the smallest discourse units are single sentences, as argued by Joty et al. (2013), it is not feasible to use a long chain to represent all constituents, due to the fact that it takes $O(TM^2)$ time to perform the forward-backward exact inference on a chain with T units and an output vocabulary size of M, thus the overall complexity for all possible sequences in their model is $O(M^2n^3)^2$.

Instead, we choose to take a sliding-window approach to form CRF chains for a particular pair of constituents, as shown in Figure 4b. For example, suppose we wish to compute the structural probability for the pair U_{j-1} and U_j, we form three chains, each of which contains two contextual constituents: $C^1 = \{U_{j-3}, U_{j-2}, U_{j-1}, U_j\}$, $C^2 = \{U_{j-2}, U_{j-1}, U_j, U_{j+1}\}$, and $C^3 = \{U_{j-1}, U_j, U_{j+1}, U_{j+2}\}$. We then find the chain $C^t, 1 \leq t \leq 3$, with the highest joint probability over the entire sequence, and assign its marginal probability $P(S_j^t = 1)$ to $P(S_j = 1)$.

Similar to M_{intra}^{struct}, for M_{multi}^{struct}, we also include additional constraints in the Viterbi decoding, by disallowing transitions between two ones, and disallowing the sequence to be all zeros if it contains all the remaining constituents in the document.

4.3 Relation models

4.3.1 Intra-sentential relation model

The intra-sentential relation model M_{intra}^{rel}, shown in Figure 5a, works in a similar way to M_{intra}^{struct}, as described in Section 4.2.1. The linear-chain CRF contains a first layer of all discourse constituents U_j's in the sentence on level i, and a second layer of relation nodes R_j's to represent the relation between a pair of discourse constituents.

However, unlike the structure model, adjacent relation nodes do not share discourse constituents on the first layer. Rather, each relation node R_j attempts to model the relation of one single constituent U_j, by taking U_j's left and right subtrees $U_{j,L}$ and $U_{j,R}$ as its first-layer nodes; if U_j is a single EDU, then the first-layer node of R_j is simply U_j, and R_j is a special relation symbol LEAF[3]. Since we know, a priori, that the constituents in the chains are either leaf nodes or the ones that have been merged by our structure model, we never need to worry about the NO-REL issue as outlined in Section 4.1.

In the bottom-up tree-building process, after merging a pair of adjacent constituents using M_{intra}^{struct} into a new constituent, say U_j, we form a chain consisting of all current constituents in the sentence to decide the relation label for U_j, i.e., the R_j node in the chain. In fact, by performing inference on this chain, we produce predictions not only for R_j, but also for all other R nodes in the chain, which correspond to all other constituents in the sentence. Since those non-leaf constituents are already labeled in previous steps in the tree-building, we can now **re-assign** their relations if the model predicts differently in this step. Therefore, this re-labeling procedure can compensate for the loss of accuracy caused by our greedy bottom-up strategy to some extent.

4.3.2 Multi-sentential relation model

Figure 5b shows our multi-sentential relation model. Like M_{intra}^{rel}, the first layer consists of adjacent discourse units, and the relation nodes on the second layer model the relation of each constituent separately.

Similar to M_{multi}^{struct} introduced in Section 4.2.2, M_{multi}^{rel} also takes a sliding-window approach to predict labels for constituents in a local context. For a constituent U_j to be predicted, we form three chains, and use the chain with the highest joint probability to assign or re-assign relations to constituents in that chain.

---

[2]The time complexity will be reduced to $O(M^2n^2)$, if we use the same chain for all constituents as in our M_{intra}^{struct}.

[3]These leaf constituents are represented using a special feature vector `is_leaf = True`; thus the CRF never labels them with relations other than LEAF.
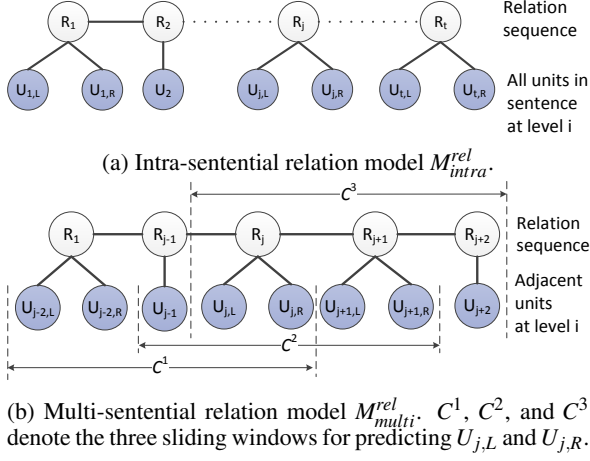
(a) Intra-sentential relation model M_{intra}^{rel}.

(b) Multi-sentential relation model M_{multi}^{rel}. C^1, C^2, and C^3 denote the three sliding windows for predicting $U_{j,L}$ and $U_{j,R}$.

Figure 5: Local relation models.

5 Post-editing

After an intra- or multi-sentential discourse tree is fully built, we perform a post-editing to consider possible modifications to the current tree, by considering useful information from the discourse constituents on upper levels, which is unavailable in the bottom-up tree-building process.

The motivation for post-editing is that, some particular discourse relations, such as TEXTUAL-ORGANIZATION, tend to occur on the top levels of the discourse tree; thus, information such as the depth of the discourse constituent can be quite indicative. However, the exact depth of a discourse constituent is usually unknown in the bottom-up tree-building process; therefore, it might be beneficial to modify the tree by including top-down information after the tree is fully built.

The process of post-editing is shown in Algorithm 1. For each input discourse tree T, which is already fully built by bottom-up tree-building models, we do the following:

Lines 3 – 9: Identify the lowest level of T on which the constituents can be modified according to the post-editing structure component, P^{struct}. To do so, we maintain a list L to store the discourse constituents that need to be examined. Initially, L consists of all the bottom-level constituents in T. At each step of the loop, we consider merging the pair of adjacent units in L with the highest probability predicted by P^{struct}. If the predicted pair is not merged in the original tree T, then a possible modification is located; otherwise, we merge the pair, and proceed to the next iteration.

Lines 10 – 12: If modifications have been proposed in the previous step, we build a new tree

---

Algorithm 1 Post-editing algorithm.

Input: A fully built discourse tree T.
1: **if** $|T| = 1$ **then**
2:     **return** T ▷ Do nothing if it is a single EDU.
3:    $L \leftarrow [U_1, U_2, \ldots, U_t]$ ▷ The bottom-level constituents in T.
4: **while** $|L| > 2$ **do**
5:     $i \leftarrow$ PREDICTMERGING(L, P^{struct})
6:     $p \leftarrow$ PARENT$(L[i], L[i+1], T)$
7:     **if** $p =$ NULL **then**
8:        **break**
9:     Replace $L[i]$ and $L[i+1]$ with p
10: **if** $|L| = 2$ **then**
11:    $L \leftarrow [U_1, U_2, \ldots, U_t]$
12: $T^p \leftarrow$ BUILDTREE$(L, P^{struct}, P^{rel}, T)$
Output: T^p

T^p using P^{struct} as the structure model, and P^{rel} as the relation model, from the constituents on which modifications are proposed. Otherwise, T^p is built from the bottom-level constituents of T. The upper-level information, such as the depth of a discourse constituent, is derived from the initial tree T.

5.1 Local models

The local models, $P_{\{intra|multi\}}^{\{struct|rel\}}$, for post-editing is almost identical to their counterparts of the bottom-up tree-building, except that the linear-chain CRFs in post-editing includes additional features to represent information from constituents on higher levels (to be introduced in Section 7).

6 Linear time complexity

Here we analyze the time complexity of each component in our discourse parser, to quantitatively demonstrate the time efficiency of our model. The following analysis is focused on the bottom-up tree-building process, but a similar analysis can be carried out for the post-editing process. Since the number of operations in the post-editing process is roughly the same (1.5 times in the worst case) as in the bottom-up tree-building, post-editing shares the same complexity as the tree-building.

6.1 Intra-sentential parsing

Suppose the input document is segmented into n sentences, and each sentence S_k contains m_k EDUs. For each sentence S_k with m_k EDUs, the

overall time complexity to perform intra-sentential parsing is $O(m_k^2)$. The reason is the following. On level i of the bottom-up tree-building, we generate a single chain to represent the structure or relation for all the $m_k - i$ constituents that are currently in the sentence. The time complexity for performing forward-backward inference on the single chain is $O((m_k - i) \times M^2) = O(m_k - i)$, where the constant M is the size of the output vocabulary. Starting from the EDUs on the bottom level, we need to perform inference for one chain on each level during the bottom-up tree-building, and thus the total time complexity is $\Sigma_{i=1}^{m_k} O(m_k - i) = O(m_k^2)$.

The total time to generate sentence-level discourse trees for n sentences is $\Sigma_{k=1}^{n} O(m_k^2)$. It is fairly safe to assume that each m_k is a constant, in the sense that m_k is independent of the total number of sentences in the document. Therefore, the total time complexity $\Sigma_{k=1}^{n} O(m_k^2) \leq n \times O(\max_{1 \leq j \leq n}(m_j^2)) = n \times O(1) = O(n)$, i.e., linear in the total number of sentences.

6.2 Multi-sentential parsing

For multi-sentential models, M_{multi}^{struct} and M_{multi}^{rel}, as shown in Figures 4b and 5b, for a pair of constituents of interest, we generate multiple chains to predict the structure or the relation.

By including a constant number k of discourse units in each chain, and considering a constant number l of such chains for computing each adjacent pair of discourse constituents ($k = 4$ for M_{multi}^{struct} and $k = 3$ for M_{multi}^{rel}; $l = 3$), we have an overall time complexity of $O(n)$. The reason is that it takes $l \times O(kM^2) = O(1)$ time, where l, k, M are all constants, to perform exact inference for a given pair of adjacent constituents, and we need to perform such computation for all $n - 1$ pairs of adjacent sentences on the first level of the tree-building. Adopting a greedy approach, on an arbitrary level during the tree-building, once we decide to merge a certain pair of constituents, say U_j and U_{j+1}, we only need to recompute a small number of chains, i.e., the chains which originally include U_j or U_{j+1}, and inference on each chain takes $O(1)$. Therefore, the total time complexity is $(n - 1) \times O(1) + (n - 1) \times O(1) = O(n)$, where the first term in the summation is the complexity of computing all chains on the bottom level, and the second term is the complexity of computing the constant number of chains on higher levels.

We have thus showed that the time complexity is *linear* in n, which is the number of sentences in the document. In fact, under the assumption that the number of EDUs in each sentence is independent of n, it can be shown that the time complexity is also linear in the total number of EDUs[4].

7 Features

In our local models, to encode two adjacent units, U_j and U_{j+1}, within a CRF chain, we use the following 10 sets of features, some of which are modified from Joty et al.'s model.

Organization features: Whether U_j (or U_{j+1}) is the first (or last) constituent in the sentence (for intra-sentential models) or in the document (for multi-sentential models); whether U_j (or U_{j+1}) is a bottom-level constituent.

Textual structure features: Whether U_j contains more sentences (or paragraphs) than U_{j+1}.

N-gram features: The beginning (or end) lexical n-grams in each unit; the beginning (or end) POS n-grams in each unit, where $n \in \{1, 2, 3\}$.

Dominance features: The PoS tags of the head node and the attachment node; the lexical heads of the head node and the attachment node; the dominance relationship between the two units.

Contextual features: The feature vector of the previous and the next constituent in the chain.

Substructure features: The root node of the left and right discourse subtrees of each unit.

Syntactic features: whether each unit corresponds to a single syntactic subtree, and if so, the top PoS tag of the subtree; the distance of each unit to their lowest common ancestor in the syntax tree (intra-sentential only).

Entity transition features: The type and the number of entity transitions across the two units. We adopt Barzilay and Lapata (2008)'s entity-based local coherence model to represent a document by an entity grid, and extract local transitions among entities in continuous discourse constituents. We use bigram and trigram transitions with syntactic roles attached to each entity.

---

[4]We implicitly made an assumption that the parsing time is dominated by the time to perform inference on CRF chains. However, for complex features, the time required for feature computation might be dominant. Nevertheless, a careful caching strategy can accelerate feature computation, since a large number of multi-sentential chains overlap with each other.

Cue phrase features: Whether a cue phrase occurs in the first or last EDU of each unit. The cue phrase list is based on the connectives collected by Knott and Dale (1994)

Post-editing features: The depth of each unit in the initial tree.

8 Experiments

For pre-processing, we use the Stanford CoreNLP (Klein and Manning, 2003; de Marneffe et al., 2006; Recasens et al., 2013) to syntactically parse the texts and extract coreference relations, and we use Penn2Malt[5] to lexicalize syntactic trees to extract dominance features.

For local models, our structure models are trained using MALLET (McCallum, 2002) to include constraints over transitions between adjacent labels, and our relation models are trained using CRFSuite (Okazaki, 2007), which is a fast implementation of linear-chain CRFs.

The data that we use to develop and evaluate our discourse parser is the RST Discourse Treebank (RST-DT) (Carlson et al., 2001), which is a large corpus annotated in the framework of RST. The RST-DT consists of 385 documents (347 for training and 38 for testing) from the *Wall Street Journal*. Following previous work on the RST-DT (Hernault et al., 2010; Feng and Hirst, 2012; Joty et al., 2012; Joty et al., 2013), we use 18 coarse-grained relation classes, and with nuclearity attached, we have a total set of 41 distinct relations. Non-binary relations are converted into a cascade of right-branching binary relations.

9 Results and Discussion

9.1 Parsing accuracy

We compare four different models using manual EDU segmentation. In Table 1, the *j*CRF model in the first row is the optimal CRF model proposed by Joty et al. (2013). *g*SVM^{FH} in the second row is our implementation of HILDA's greedy parsing algorithm using Feng and Hirst (2012)'s enhanced feature set. The third model, *g*CRF, represents our greedy CRF-based discourse parser, and the last row, *g*CRF^{PE}, represents our parser with the post-editing component included.

In order to conduct a direct comparison with Joty et al.'s model, we use the same set of eval-

Model	Span	Nuc	Relation	
			Acc	MAFS
*j*CRF	82.5	68.4	55.7	N/A
*g*SVM^{FH}	82.8	67.1	52.0	27.4/23.3
*g*CRF	84.9*	69.9*	57.2*	35.3/31.3
*g*CRF^{PE}	**85.7***†	**71.0***†	**58.2***†	**36.2/32.3**
Human	88.7	77.7	65.8	N/A

∗: significantly better than *g*SVM^{FH} ($p < .01$)
†: significantly better than *g*CRF ($p < .01$)

Table 1: Performance of different models using gold-standard EDU segmentation, evaluated using the constituent accuracy (%) for span, nuclearity, and relation. For relation, we also report the macro-averaged F1-score (MAFS) for correctly retrieved constituents (before the slash) and for all constituents (after the slash). Statistical significance is verified using Wilcoxon's signed-rank test.

uation metrics, i.e., the unlabeled and labeled precision, recall, and F-score[6] as defined by Marcu (2000). For evaluating relations, since there is a skewed distribution of different relation types in the corpus, we also include the macro-averaged F1-score (MAFS)[7] as another metric, to emphasize the performance of infrequent relation types. We report the MAFS separately for the correctly retrieved constituents (i.e., the span boundary is correct) and all constituents in the reference tree.

As demonstrated by Table 1, our greedy CRF models perform significantly better than the other two models. Since we do not have the actual output of Joty et al.'s model, we are unable to conduct significance testing between our models and theirs. But in terms of overall accuracy, our *g*CRF model outperforms their model by 1.5%. Moreover, with post-editing enabled, *g*CRF^{PE} significantly ($p < .01$) outperforms our initial model *g*CRF by another 1% in relation assignment, and this overall accuracy of 58.2% is close to 90% of human performance. With respect to the macro-averaged F1-scores, adding the post-editing component also obtains about 1% improvement.

However, the overall MAFS is still at the lower

---

[6] For manual segmentation, precision, recall, and F-score are the same.

[7] MAFS is the F1-score averaged among all relation classes by equally weighting each class. Therefore, we cannot conduct significance test between different MAFS.

	Avg	Min	Max
# of EDUs	61.74	4	304
# of Sentences	26.11	2	187
# of EDUs per sentence	2.36	1	10

Table 2: Characteristics of the 38 documents in the test data.

end of 30% for all constituents. Our error analysis shows that, for two relation classes, TOPIC-CHANGE and TEXTUAL-ORGANIZATION, our model fails to retrieve any instance, and for TOPIC-COMMENT and EVALUATION, our model scores a class-wise F_1 score lower than 5%. These four relation classes, apart from their infrequency in the corpus, are more abstractly defined, and thus are particularly challenging.

9.2 Parsing efficiency

We further illustrate the efficiency of our parser by demonstrating the time consumption of different models.

First, as shown in Table 2, the average number of sentences in a document is 26.11, which is already too large for optimal parsing models, e.g., the CKY-like parsing algorithm in *j*CRF, let alone the fact that the largest document contains several hundred of EDUs and sentences. Therefore, it should be seen that non-optimal models are required in most cases.

In Table 3, we report the parsing time[8] for the last three models, since we do not know the time of *j*CRF. Note that the parsing time excludes the time cost for any necessary pre-processing. As can be seen, our *g*CRF model is considerably faster than *g*SVM^{FH}, because, on one hand, feature computation is expensive in *g*SVM^{FH}, since *g*SVM^{FH} utilizes a rich set of features; on the other hand, in *g*CRF, we are able to accelerate decoding by multi-threading MALLET (we use four threads). Even for the largest document with 187 sentences, *g*CRF is able to produce the final tree after about 40 seconds, while *j*CRF would take over 16 hours assuming each DCRF decoding takes only 0.01 second. Although enabling post-editing doubles the time consumption, the overall time is still acceptable in practice, and the loss of efficiency can be compensated by the improvement in accuracy.

[8]Tested on a Linux system with four duo-core 3.0GHz processors and 16G memory.

Model	Parsing Time (seconds)		
	Avg	Min	Max
*g*SVM^{FH}	11.19	0.42	124.86
*g*CRF	5.52	0.05	40.57
*g*CRF^{PE}	10.71	0.12	84.72

Table 3: The parsing time (in seconds) for the 38 documents in the test set of RST-DT. Time cost of any pre-processing is excluded from the analysis.

10   Conclusions

In this paper, we presented an efficient text-level discourse parser with time complexity linear in the total number of sentences in the document. Our approach was to adopt a greedy bottom-up tree-building, with two linear-chain CRFs as local probabilistic models, and enforce reasonable constraints in the first CRF's Viterbi decoding. While significantly outperforming the state-of-the-art model by Joty et al. (2013), our parser is much faster in practice. In addition, we propose a novel idea of post-editing, which modifies a fully-built discourse tree by considering information from upper-level constituents. We show that, although doubling the time consumption, post-editing can further boost the parsing performance to close to 90% of human performance.

In future work, we wish to further explore the idea of post-editing, since currently we use only the depth of the subtrees as upper-level information. Moreover, we wish to study whether we can incorporate constraints into the relation models, as we do to the structure models. For example, it might be helpful to train the relation models using additional criteria, such as Generalized Expectation (Mann and McCallum, 2008), to better take into account some prior knowledge about the relations. Last but not least, as reflected by the low MAFS in our experiments, some particularly difficult relation types might need specifically designed features for better recognition.

Acknowledgments

References

Jason Baldridge and Alex Lascarides. 2005. Probabilistic head-driven parsing for discourse structure. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 96–103, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: an entity-based approach. *Computational Linguistics*, 34(1):1–34.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. In *Proceedings of Second SIGDial Workshop on Discourse and Dialogue (SIGDial 2001)*, pages 1–10.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*.

Vanessa Wei Feng and Graeme Hirst. 2012. Text-level discourse parsing with rich linguistic features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL 2012)*, pages 60–68, Jeju, Korea.

Hugo Hernault, Helmut Prendinger, David A. duVerle, and Mitsuru Ishizuka. 2010. HILDA: A discourse parser using support vector machine classification. *Dialogue and Discourse*, 1(3):1–33.

Shafiq Joty, Giuseppe Carenini, and Raymond T. Ng. 2012. A novel discriminative framework for sentence-level discourse analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL 2012, pages 904–915.

Shafiq Joty, Giuseppe Carenini, Raymond Ng, and Yashar Mehdad. 2013. Combining intra- and multi-sentential rhetorical parsing for document-level discourse analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 486–496, Sofia, Bulgaria, August.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL 2003)*, ACL 2003, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alistair Knott and Robert Dale. 1994. Using linguistic phenomena to motivate a set of coherence relations. *Discourse Processes*, 18(1):35–64.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML 2001, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Gideon S. Mann and Andrew McCallum. 2008. Generalized Expectation Criteria for semi-supervised learning of Conditional Random Fields. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL 2008)*, pages 870–878, Columbus, Ohio, June. Association for Computational Linguistics.

William Mann and Sandra Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Daniel Marcu. 2000. *The Theory and Practice of Discourse Parsing and Summarization*. The MIT Press.

Andrew Kachites McCallum. 2002. MALLET: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Philippe Muller, Stergos Afantenos, Pascal Denis, and Nicholas Asher. 2012. Constrained decoding for text-level discourse parsing. In *Proceedings of COLING 2012*, pages 1883–1900, Mumbai, India, December. The COLING 2012 Organizing Committee.

Naoaki Okazaki. 2007. CRFsuite: a fast implementation of conditional random fields (CRFs). http://www.chokkan.org/software/crfsuite/.

Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. 2013. The life and death of discourse entities: Identifying singleton mentions. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 627–633, Atlanta, Georgia, June. Association for Computational Linguistics.

Rajen Subba and Barbara Di Eugenio. 2009. An effective discourse parser that uses rich linguistic information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 566–574, Boulder, Colorado, June. Association for Computational Linguistics.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *The Journal of Machine Learning Research*, 8:693–723, May.

Christian Wojek and Bernt Schiele. 2008. A dynamic conditional random field model for joint labeling of object and scene classes. In *European Conference*

on Computer Vision (ECCV 2008), pages 733–747, Marseille, France.

Dong Yang, Paul Dixon, Yi-Cheng Pan, Tasuku Oonishi, Masanobu Nakamura, and Sadaoki Furui. 2009. Combining a two-step conditional random field model and a joint source channel model for machine transliteration. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 72–75, Suntec, Singapore, August. Association for Computational Linguistics.