A Sequential Model for Discourse Segmentation

Hugo Hernault, Danushka Bollegala, and Mitsuru Ishizuka

Graduate School of Information Science and Technology,
The University of Tokyo 7-3-1 Hongo,
Bunkyo-ku, Tokyo 113-8656, Japan
hugo@mi.ci.i.u-tokyo.ac.jp, danushka@mi.ci.i.u-tokyo.ac.jp,
ishizuka@i.u-tokyo.ac.jp

Abstract. Identifying discourse relations in a text is essential for various tasks in Natural Language Processing, such as automatic text summarization, question-answering, and dialogue generation. The first step of this process is segmenting a text into elementary units. In this paper, we present a novel model of discourse segmentation based on sequential data labeling. Namely, we use Conditional Random Fields to train a discourse segmenter on the RST Discourse Treebank, using a set of lexical and syntactic features. Our system is compared to other statistical and rule-based segmenters, including one based on Support Vector Machines. Experimental results indicate that our sequential model outperforms current state-of-the-art discourse segmenters, with an F-score of 0.94. This performance level is close to the human agreement F-score of 0.98.

1 Introduction

Discourse structures have an important role in various computational tasks, such as creating text summaries [1], performing question-answering [2], generating dialogues [3], or improving the processing of clinical guidelines [4]. For example, in automatic text summarization, if we know that a particular segment of text further elaborates an already stated fact, then we can safely ignore the elaborated segment to create a concise summary of the text. However, despite the wide uses of discourse parsing, true automatization of these tasks is preconditioned by the availability of efficient discourse parsers. In the past few years, several research efforts have aimed at building automatic parsers. In particular, a number of authors have attempted to create discourse parsers in the framework of the Rhetorical Structure Theory (RST) [5], one of the most prevalent discourse theories.

The general problem of automatically annotating a text with a set of discourse relations can be decomposed into three sub-problems. First, the text is divided into non-overlaping units, called *elementary discourse units* (EDUs). Each discourse theory has its own specificities in terms of segmentation guidelines and size of units. In RST, units are essentially clauses, and a sentence may be segmented in the fashion of Figure 1.

Second, we must select, from a pre-defined set, which discourse relations hold between consecutive pairs of EDUs. In previous work on discourse tagging, this

[The posters were printed on paper]^{1A} [pre-signed by Mr. Dali,]^{1B} [the attorneys said.]^{1C} (wsj$_{1331}$)

Fig. 1. A sentence split into three EDUs

problem has been modeled as a supervised classification task: a multi-class classifier is trained to identify the discourse relation holding between two given EDUs [6]. Last, we must construct a single discourse tree depicting the way all discourse relations and EDUs of the text relate to each-other. This paper focuses on the first sub-problem, segmenting a given text into a sequence of EDUs.

The overall accuracy of discourse parsing depends on the initial segmentation step. Indeed, if the text is wrongly segmented during this first stage, it becomes unreliable to assign correct discourse relations or build a consistent discourse tree for the text [7]. Therefore, the discourse segmentation task is of paramount importance to any discourse parsing algorithm.

As described later in Section 1.1, existing discourse segmentation algorithms either use a set of hand-coded rules or a supervised classifier. Considering the heterogenous texts that must be processed by a discourse parser, it is not feasible to write rules to cover every segmentation criteria. Thus, modeling the discourse segmentation problem as a classification task [8], and training a classification model from human-annotated data, is an interesting solution to the problems encountered with rule-based discourse segmentation. In the classification approach, given a word and its context in the text, the system determines whether there is likely an EDU boundary. Commonly-used features include punctuation and cue phrases (e.g., *but*, *and*, *however*). However, this method does not consider prior segmentation decisions as it encounters a new word. Each word is considered individually by the classifier. Therefore, the decisions made by a classification approach are locally-motivated. We propose a sequence labeling approach to discourse segmentation, that finds the globally optimum segmentation into EDUs for a given text. In contrast to the classification approach to segmentation, the model proposed in this paper considers its own previous decisions before it determines whether it should impose a discourse boundary.

1.1 Related Work

In 'SPADE' [7], a segmenter based on a probabilistic model is implemented, as the first step to a sentence-level discourse parser. This classifier is trained on the RST Discourse Treebank corpus (RST-DT) [9], and then used to differentiate between EDU boundary and non-boundary words. The boundary probability is calculated by counting occurrences of certain lexico-syntactic patterns in the training corpus. The segmenter yields an F-score of 0.831 (0.847 when using perfect parse trees). Although the features used by the authors are very relevant, the model shows its limits when compared to more elaborated probabilistic approaches.

A statistical discourse segmenter, based on artificial neural networks, is presented in [10]. The system is also trained on the RST-DT, and uses syntactic and lexical information, in particular discourse cues. A multilayer perceptron model is employed, and bagging is used in order to reduce overfitting. The performance of the segmenter is comparable to [7], with an F-score of 0.844 (0.860 when using perfect parse trees).

Rule-based discourse segmenters have also been created: In [11], segmentation is done by a multi-step algorithm, which uses syntactic information and discourse cues. An F-score of 0.80 is reported for this system. Then, in [12], it is argued that using rules has certain advantages over automatic learning methods. Indeed, the proposed model does not depend on a specific training corpus, and allows for high-precision segmentation, as it inserts fewer but 'quality' boundaries. However, in the latter system, segmentation is done in a manner different from the segmentation guidelines used in the RST-DT corpus. First, the authors avoid building short EDUs, in order to avoid relations with lesser informative content, such as SAME-UNIT or ELABORATION. Then, complement clauses are not placed in autonomous units. For instance, *'He said that'* is not considered an EDU. In this paper, we will not enter the discussion of what constitutes the best segmentation guidelines, and focus instead on the supervised methods for learning segmentation efficiently from the RST-DT corpus.

We already pointed out that discourse segmentation is necessary in order to build an automatic discourse parser. It is however interesting to note that the conception of a discourse relation analyzer is possible without treating the segmentation problem. In [6], a discourse parser using a multi-class Support Vector Machines classifier for relation identification, and a greedy bottom-up tree-building algorithm is described. The algorithm is first evaluated on perfectly-segmented EDUs from the RST-DT. Then, in order to create a fully automatic system, the authors build on top of the discourse segmenter of [7]. When using perfect segmentation, the discourse parsing pipeline returns a F-score of 0.548, but this score drops to 0.443 when using SPADE's segmenter. Here, we clearly see the critical role of the segmentation component, which can easily act as a bottleneck, and pull down the performance of the whole system.

Finally, it is worthy to note that the study of discourse segmentation, although seemingly restricted in its scope, can potentially be beneficial to all applications in which discourse relations or discourse structures are used.

2 Method

2.1 Outline

We model the problem of discourse segmentation as a sequential labeling task. Although there have been classifier-based approaches to discourse segmentation such as neural network-based methods [10], to our knowledge, this is the first attempt to model discourse segmentation as a sequential labeling problem. To illustrate the proposed sequential labeling approach, let us consider the example sentence shown in Figure 2. Therein, BOS and EOS respectively denote the

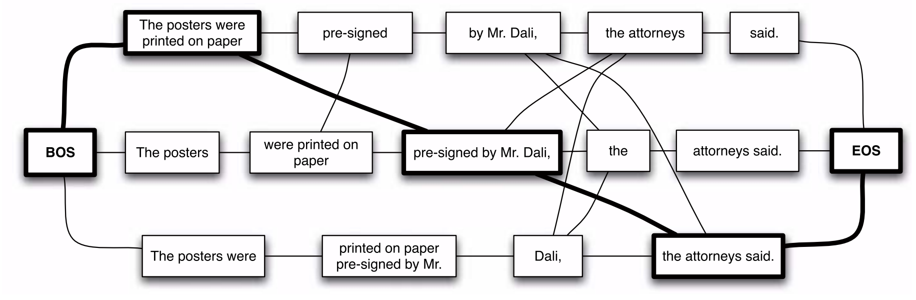

Fig. 2. Some possible segmentation decisions

beginning of the sentence and the *end of the sentence*. Because EDUs are not allowed to cross sentence boundaries, BOS and EOS act as segment boundaries. There are multiple possible ways in which we can segment the text inside a sentence. These different ways can be represented by a lattice structure, like the one shown in Figure 2. Then, the problem of segmenting a given sentence into EDUs can be seen as searching for the best path connecting BOS to EOS on this lattice. This best path is the one that maximizes the likelihood – alternatively, we can consider minimizing a cost function – of the segmentation process. Modeling a sentence as a lattice and then searching for the best path is a technique used in various related tasks in Natural Language Processing, such as part-of-speech (POS) tagging, chunking, and named-entity recognition (NER). The best segmentation path in our example sentence is shown in bold in Figure 2.

In Section 2.2, we introduce Conditional Random Fields [13], the sequential labeling algorithm that we use to find the best path on the lattice. CRFs have shown to outperform other sequential labeling algorithms such as Hidden Markov Models (HMMs) and Maximum Entropy Markov Models (MEMMs), on a wide range of tasks including POS tagging, chunking and NER [13]. In Section 2.3 we describe how we employ CRFs for the task of discourse segmentation. The features that we use for training are described in Section 2.4.

2.2 Conditional Random Fields

Conditional Random Fields or Markov Random Fields are undirected graphical models that express the relationship between some random variables. Each vertex in the undirected graph represents a random variable. Some of those variables might be observable (e.g., the frequency of a particular word), whereas other variables cannot be observed (i.e., the POS tag of the word). In Conditional Random Fields, by definition each random variable must obey the Markov property with respect to the graph (i.e., each node is independent from the other nodes, when conditioned upon its Markov blanket). In the general setting where the Conditional Random Field represents an arbitrary undirected graph, each clique in the graph is assigned with a potential function. However, in the sequence labeling task, the undirected graph reduces to a linear chain, as the one shown in Figure 2.2. In

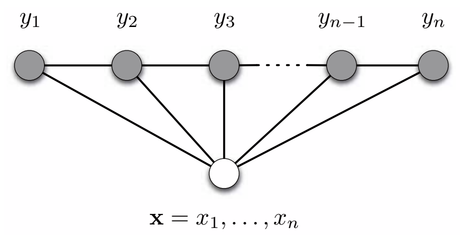

Fig. 3. Graphical model of a linear-chain CRF

Figure 2.2, the hidden variables are shown in shaded circles, whereas observed variables are shown in white. From the Markov assumption, it follows that each hidden variable only depends on its neighboring nodes.

Log-linear models have been used as potential functions in CRFs for their convenience in computation. Given an input observation $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathcal{X}^n$, CRF computes the probability $p(\boldsymbol{y}|\boldsymbol{x})$ of a possible output $\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathcal{Y}^n$. The general formulation of the linear-chain CRF is,

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_i f_i(y_{j-1}, y_j, x_j)\right). \tag{1}$$

Here, $f_i(y_{j-1}, y_j, \boldsymbol{x})$ is a binary-valued feature that returns the value 1 if the corresponding feature is fired when moving from the hidden state y_{j-1} to y_j, after observing x. For example, we can define a feature that encodes the property *previous position is not an* EDU *boundary and the current token is a comma.* λ_i is the weight associated with feature f_i. The linear sum of weights over features is taken inside the exponential and the normalization constant Z (also known as the partition function) is set such that the sum of probabilities over all possible label sequences, $\mathfrak{Y}(\boldsymbol{x})$, for $\boldsymbol{x}$, equals one. The expression of the normalization constant is,

$$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathfrak{Y}(x)} \exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_i f_i(y_{j-1}, y_j, x_j)\right). \tag{2}$$

We can further simplify the notation by introducing the global feature vector, $\boldsymbol{F}(\boldsymbol{y}, \boldsymbol{x}) = \{F_1(\boldsymbol{y}, \boldsymbol{x}), \ldots, F_m(\boldsymbol{y}, \boldsymbol{x})\}$, where $F_i(\boldsymbol{y}, \boldsymbol{x}) = \sum_{j=1}^{N} f_i(y_{j-1}, y_j, x_j)$. Moreover, we define the weight vector $\varLambda = \{\lambda_1, \ldots, \lambda_m\}$. Then, Equation 1 can be written as,

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp(\varLambda \cdot \boldsymbol{F}(\boldsymbol{y}, \boldsymbol{x})). \tag{3}$$

Here, $\cdot$ denotes the inner-product between the global feature vector and weight vector.

Given a training dataset, $T = \{(\boldsymbol{x_j}, \boldsymbol{y_j})\}_{j=1}^{N}$, the weights λ_i are computed such that the log-likelihood L_Λ of T is maximized. Log-likelihood over the training dataset can be written as follows,

$$\begin{aligned} L_\Lambda &= \sum_{j=1}^{N} \log(P(y_j|x_j)) \\ &= \sum_{j=1}^{N} \left(\sum_{\boldsymbol{y} \in \mathfrak{Y}(x_j)} \exp(\Lambda \cdot (\boldsymbol{F}(y_j, x_j) - \boldsymbol{F}(y, x_j))) \right) \\ &= \sum_{j=1}^{N} \left(\Lambda \cdot \boldsymbol{F}(y_j, x_j) - \log(Z_{x_j}) \right) . \end{aligned}$$

Therefore, to maximize the log-likelihood, we must maximize the difference between the inner-products of the correct labelling $\Lambda \cdot \boldsymbol{F}(y_j, x_j)$ and all other candidate labellings $\Lambda \cdot \boldsymbol{F}(y, x_j)$ for $y \in \mathfrak{Y}(x_j)$. In practice, to avoid overfitting the weights to the training data, the vector Λ is regularized. Two popular choices for vector regularization are L_1 and L_2 regularizations. In general, the $\mathrm{L}_\mathrm{k}(\boldsymbol{x})$ regularization of an n-dimensional vector $\boldsymbol{x}$ is defined as,

$$\mathrm{L}_\mathrm{k}(\boldsymbol{x}) = \sqrt[k]{\sum_{i=1}^{n} {x_i}^k}. \tag{4}$$

The final optimization function $\mathrm{H}(\Lambda)$ with regularization can be written as,

$$\mathrm{H}(\Lambda) = L_\Lambda - \sigma L_k. \tag{5}$$

Here, σ is a regularization coefficient that determines the overall effect of regularization towards training. This optimization problem can be efficiently solved by using gradient descent algorithms. We used CRFsuite [14] with L_1 regularization, which implements the orthant-wise limited memory quasi-Newton (OWL-QN) method. The regularization coefficient is set to its default value of 1.

Because CRFs are discriminative models, they capture many correlated features of the input. This property of CRFs is particularly useful, because we can define as many features as we like, irrespective of whether those features are mutually independent or not. If a particular feature is not useful, then it will be assigned a lower weight and will effectively get removed in the final trained model. In particular, L_1 regularization yields sparser models compared to L_2, thereby removing many useless features automatically [15].

Once trained, the CRF can then be used to find the optimal labeling sequence $\hat{\boldsymbol{y}}$ for a given input $\boldsymbol{x}$ as,

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y} \in \mathfrak{Y}(\boldsymbol{x})} p(\boldsymbol{y}|\boldsymbol{x}). \tag{6}$$

2.3 Application to Discourse Segmentation

In the case of discourse segmentation, the problem is to assign each word in the input text an observation category $c \in \{C, B\}$, where B denotes a *beginning* of EDU, and C a *continuation* of EDU. For example, given the snippet from Figure 1:

```
The posters were printed on paper pre-signed by Mr. Dali ,
the attorneys said .
```

The output sequence is the following:

```
B C C C C C B C C C C B C C C
```

2.4 Features

We use a combination of syntactic and lexical features: words, POS tags, lexical heads. In particular, we use the lexico-syntactic features of [7], which were found to constitute a good indication of the presence of EDU boundaries.

Figure 2.4 shows part of the sentence's syntax tree, in which lexical heads have also been indicated, using projection rules from [16]. For a word w, we look at its highest ancestor in the parse tree with a lexical head equal to w, and with a right-sibling. We call this highest-ancestor node N_w, N_p its parent, and N_r its right-sibling. For instance, when following this process for the word 'paper', we get N_w = NP(paper), N_p = NP(paper), N_r = VP(pre-signed).

We define as *contextual features at position i* in the text, the set composed of the word w_i, its POS, as well as the POS and lexical heads of N_{wi}, N_{p_i}, and N_{ri}. In the experiments of Section 3, unless stated otherwise, the features for position i in the text are created by concatenating the contextual features at positions

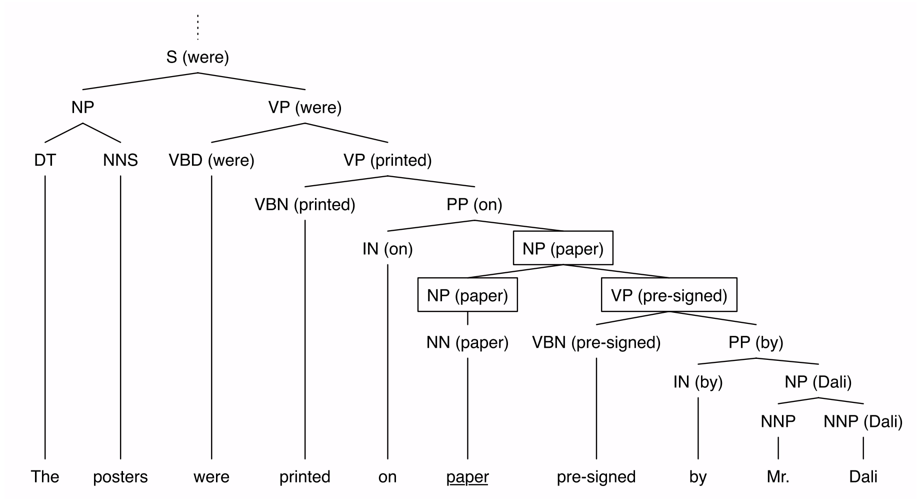

Fig. 4. Partial lexicalized syntax tree

$i-2$, $i-1$, and i. These elements are then encoded into feature templates. For instance, the template encoding the property *the current word is 'paper'* is,

$$g(\boldsymbol{x}, j) = \begin{cases} 1 \text{ if } x_j = \texttt{paper} \\ 0 \text{ otherwise} \end{cases} .$$

These templates are used as a basis to generate the CRF feature functions defined in Section 2.2. Our working corpus, the RST-DT, contains 385 texts from the Wall Street Journal (347 for training, 38 as a test subset). After feature extraction, we obtain 177,633 training vectors and 21,667 test vectors.

3 Experiments

We first implement our CRF-based segmenter, referred to as 'CRFSeg' in the rest of the paper. Three versions of the segmenter are created, using parse trees from different sources. First, we use trees from the Penn Treebank [17], which are gold-standard, human-annotated syntax trees. Then, we use trees generated by Charniak's syntax parser [18]. Last, we use trees generated by the Stanford parser [19].

Evaluation is done on the test subset of the RST-DT. We use the metric commonly agreed by most authors ([7], [12]), i.e., we only evaluate intra-sentence boundaries. Thus, the score is not artificially boosted by including obvious end-of-sentence or start-of-sentence boundaries. For instance, the sentence of Figure 1 is made of three EDUs, but we only take into account two boundaries. The performance of our segmenter is reported in Table 1.

As expected, using gold-standard parse trees from the Penn Treebank yields the best results, with an F-score of 0.953. Using syntax parsers instead produces slightly lower scores, particularly in terms of recall for the B label. However, the macro-average scores are almost identical for both software, with an F-score

Table 1. Detailed performance of CRFSeg

Trees	Label	Precision	Recall	F-score
Penn	B	0.927	0.897	0.912
	C	0.992	0.995	0.993
	Macro-average	0.960	0.946	0.953
Charniak	B	0.915	0.876	0.895
	C	0.990	0.993	0.992
	Macro-average	0.952	0.935	0.943
Stanford	B	0.910	0.872	0.890
	C	0.990	0.993	0.991
	Macro-average	0.950	0.932	0.941

Table 2. Performance comparison with other segmenters

System	Trees	Precision	Recall	F-score
SPADE	Penn	0.841	0.854	0.847
NNDS	Penn	0.855	0.866	0.860
CRFSeg	Penn	**0.960**	**0.946**	**0.953**
SPADE	Charniak	0.835	0.827	0.831
NNDS	Charniak	0.839	0.848	0.844
CRFSeg	Charniak	**0.952**	**0.935**	**0.943**
CRFSeg	Stanford	0.950	0.932	0.941
Human agreement	–	0.985	0.982	0.983

of 0.943 for Charniak, and 0.941 for Stanford. This suggests that the proposed method is not constrained by the choice of a specific parser.

Next, we compare the performance of our segmenter to other works. Results are presented in Table 2. NNDS indicates the Neural-Networks Discourse Segmenter [10] ; SPADE is the system described in [7]. Here, CRFSeg significantly outperforms other discourse segmenters. When using gold-standard trees, SPADE and NNDS yield respectively F-scores of 0.847 and 0.860, versus 0.953 for CRFSeg. The measure of the human annotator's agreement for the segmentation task has been calculated in [7], with a F-score of 0.983. Using CRFSeg with perfect parse trees, we reach 96.9% of this score, while we reach 95.7% of this score when using the Stanford parser.

We chose not to include in Table 2 the rule-based segmenters of [11] and [12], for several reasons. First, [11] report their results using a 'softer' metric, in which end-of-sentence boundaries are taken into account. The authors used Penn Treebank parse trees, and after evaluation on 8 texts of the RST-DT, obtain a precision of 0.814 and recall of 0.792. With the same parse trees and metric, but using the 38 texts from the standard test subset, CRFSeg obtains a precision of 0.973 and recall of 0.969. Finally, the results of [12] cannot be directly compared to ours, as different segmentation guidelines were used. The authors report there a precision of 0.890 and recall of 0.860 when using Charniak parse trees, a precision of 0.820 and recall of 0.860 when using Stanford trees. Moreover, this score is measured on 3 texts of the RST-DT only, which makes comparison even more risky.

3.1 Comparison to a Segmenter Based on Support Vector Machines

To compare the sequential model of discourse segmentation with a classification model, we implement a discourse segmenter using Support Vector Machines (SVM) [8]. SVMs have reported state-of-the-art performances in a wide range of tasks in NLP. We employ the same training data and features we previously used with CRFs, and [20] is used for the implementation. We select the RBF kernel,

Table 3. Comparison of performance, using contextual features at various positions

	CRFSeg			SVMSeg		
Positions	**Precision**	**Recall**	**F-score**	**Precision**	**Recall**	**F-score**
$(-3, -2, -1, 0)$	0.960	0.949	0.954	0.960	0.952	**0.956**
$(-2, -1, 0)$	0.960	0.946	0.953	0.965	0.954	**0.959**
$(-1, 0, 1)$	0.941	0.928	0.934	0.943	0.932	**0.938**
$(0, 1, 2)$	0.846	0.834	**0.840**	0.831	0.807	0.819
$(-1, 0)$	0.938	0.929	0.934	0.940	0.934	**0.937**
$(0, 1)$	0.843	0.830	**0.836**	0.834	0.804	0.819
(0)	0.845	0.827	**0.836**	0.821	0.801	0.811

and optimal parameters are found using grid search with 5-fold cross-validation. We dub this segmenter 'SVMSeg'.

In order to see how SVMSeg and CRFSeg perform under varied settings, we run several experiments, using contextual features from various positions. For instance, given the vector of relative positions $(-2, -1, 0)$, we build the feature vector for text position i as the concatenation of contextual features from positions $i - 2$, $i - 1$, and i. Results of the experiments with perfect parse trees are shown in Table 3.

The first striking observation is that, when using contextual features located before the current position, CRFSeg and SVMSeg perform similarly, with a slightly higher score for SVMSeg. For example, using positions $(-2, -1, 0)$, CRF-Seg and SVMSeg yield respectively F-scores of 0.953 versus 0.959, which is not a statistically significant difference. In this case, there is no clear benefit of the sequential model over a classification approach. It is also interesting to note that the cases $(-3, -2, -1, 0)$ and $(-2, -1, 0)$ produce identical results, which suggests that context that appears at a distance farther than two words from the current position is not useful for segmentation.

When using only the context of the current position, (0), CRFSeg outperforms SVMSeg (respective F-score of 0.836 versus 0.811). Here the CRF model has the upper-hand, as it is able to remember its past input data and decisions. However, including contextual features for positions ahead does not improve the score, which confirms that segmentation does not require the knowledge of future words and contexts – excepted for the interaction with the immediate next word, which is already encoded in our features, c.f. Section 2.4.

Finally, we run a head-to-head error comparison of the two models. In this experiment, we use the results from case $(-2, -1, 0)$. In order to account for all errors, the metric is changed so that we consider all EDU boundaries without restriction. Results are shown in Table 4.

We measure an error rate of 10^{-2} for CRFSeg, while SVMSeg has an error rate of $9.4 \cdot 10^{-3}$. However, 30% of the errors made by CRFSeg happen on cases where SVMSeg is correct, and reciprocally, 20% of errors made by SVMSeg occur on cases where CRFSeg is correct. A possible extension could then be to

Table 4. Comparison of errors between CRFSeg and SVMSeg

		SVMSeg		
		OK	**¬OK**	**Total**
CRFSeg	**OK**	21393	41	21434
	¬OK	70	163	233
	Total	21463	204	21667

combine both systems, and create a hybrid segmenter with a lower error rate. For instance, it is possible to measure, for each input data, the confidence of the two models, and use only the result of the model with the highest confidence. In this case, the expected error rate of the hybrid system is $7.5 \cdot 10^{-3}$ (163/21667).

4 Conclusion

We have presented a sequential model of discourse segmentation, based on Conditional Random Fields. The proposed model finds the globally optimum sequence of discourse boundaries, which makes for one of the most efficient supervised discourse segmentation methods we are aware of. Using standard automatic syntax parsers, our system reaches 96% of the human performance level. We also found that this approach performs comparably to a SVM-based discourse segmenter using contextual features. We suggested to build a hybrid system combining both models, in order to further reduce the number of incorrect segmentation decisions. In this case, we expect an error rate of $7.5 \cdot 10^{-3}$. These results validate that our segmenter is usable in a real-time discourse parsing system, in which the segmentation step is decisive for the rest of the process.

In the continuation of this work, we are currently exploring the benefits of sequential approaches for discourse relation labeling and tree construction.

References

1. Marcu, D.: The Theory and Practice of Discourse Parsing and Summarization. MIT Press, Cambridge (2000)
2. Chai, J.Y., Jin, R.: Discourse structure for context question answering. In: Harabagiu, S., Lacatusu, F. (eds.) HLT-NAACL 2004: Workshop on Pragmatics of Question Answering, Boston, Massachusetts, USA, pp. 23–30. Association for Computational Linguistics (2004)
3. Hernault, H., Piwek, P., Prendinger, H., Ishizuka, M.: Generating dialogues for virtual agents using nested textual coherence relations. In: Prendinger, H., Lester, J.C., Ishizuka, M. (eds.) IVA 2008. LNCS (LNAI), vol. 5208, pp. 139–145. Springer, Heidelberg (2008)
4. Georg, G., Hernault, H., Cavazza, M., Prendinger, H., Ishizuka, M.: From rhetorical structures to document structure: shallow pragmatic analysis for document engineering. In: DocEng 2009, pp. 185–192. ACM, New York (2009)

5. Mann, W.C., Thompson, S.A.: Rhetorical structure theory: Toward a functional theory of text organization. Text 8, 243–281 (1988)
6. du Verle, D., Prendinger, H.: A novel discourse parser based on support vector machine classification. In: ACL 2009, Suntec, Singapore, pp. 665–673. Association for Computational Linguistics (2009)
7. Soricut, R., Marcu, D.: Sentence level discourse parsing using syntactic and lexical information. In: NAACL 2003, Morristown, NJ, USA, pp. 149–156. Association for Computational Linguistics (2003)
8. Vapnik, V.N.: The nature of statistical learning theory. Springer, New York (1995)
9. Carlson, L., Marcu, D., Okurowski, M.E.: Rst discourse treebank (2002)
10. Subba, R., Di Eugenio, B.: Automatic discourse segmentation using neural networks. In: Proceedings of the 11th Workshop on the Semantics and Pragmatics of Dialogue, Trento, Italy, pp. 189–190 (2007)
11. Le, H.T., Abeysinghe, G., Huyck, C.: Automated discourse segmentation by syntactic information and cue phrases. In: AIA 2004, Innsbruck, Austria (2004)
12. Tofiloski, M., Brooke, J., Taboada, M.: A syntactic and lexical-based discourse segmenter. In: ACL 2009, Suntec, Singapore, pp. 77–80. Association for Computational Linguistics (2009)
13. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: ICML 2001, pp. 282–289. Morgan Kaufmann Publishers Inc., San Francisco (2001)
14. Okazaki, N.: Crfsuite: a fast implementation of conditional random fields, crfs (2007)
15. Ng, A.Y.: Feature selection, l1 vs. l2 regularization, and rotational invariance. In: ICML 2004, p. 78. ACM, New York (2004)
16. Magerman, D.M.: Statistical decision-tree models for parsing. In: ACL 1995, Morristown, NJ, USA, pp. 276–283. Association for Computational Linguistics (1995)
17. Marcus, M.P., Marcinkiewicz, M.A., Santorini, B.: Building a large annotated corpus of english: the penn treebank. Comput. Linguist. 19, 313–330 (1993)
18. Charniak, E.: A maximum-entropy-inspired parser. In: NAACL 2000, pp. 132–139. Morgan Kaufmann Publishers Inc., San Francisco (2000)
19. Klein, D., Manning, C.D.: Fast exact inference with a factored model for natural language parsing. In: Advances in Neural Information Processing Systems, vol. 15. MIT Press, Cambridge (2003)
20. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001), `http://www.csie.ntu.edu.tw/~cjlin/libsvm`