Flexible Text Segmentation with Structured Multilabel Classification

Ryan McDonald **Koby Crammer** **Fernando Pereira**
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
{ryantm,crammer,pereira}@cis.upenn.edu

Abstract

Many language processing tasks can be reduced to breaking the text into segments with prescribed properties. Such tasks include sentence splitting, tokenization, named-entity extraction, and chunking. We present a new model of text segmentation based on ideas from multilabel classification. Using this model, we can naturally represent segmentation problems involving overlapping and non-contiguous segments. We evaluate the model on entity extraction and noun-phrase chunking and show that it is more accurate for overlapping and non-contiguous segments, but it still performs well on simpler data sets for which sequential tagging has been the best method.

1 Introduction

Text segmentation is a basic task in language processing, with applications such as tokenization, sentence splitting, named-entity extraction, and chunking. Many parsers, translation systems, and extraction systems rely on such segmentations to accurately process the data. Depending on the application, segments may be tokens, phrases, or sentences. However, in this paper we primarily focus on segmenting sentences into tokens.

The most common approach to text segmentation is to use finite-state sequence tagging models, in which each atomic text element (character or token) is labeled with a tag representing its role in a segmentation. Models of that form include hidden Markov models (Rabiner, 1989; Bikel et al., 1999) as well as discriminative tagging models based on maximum entropy classification (Ratnaparkhi, 1996; McCallum et al., 2000), conditional random fields (Lafferty et al., 2001; Sha and Pereira, 2003), and large-margin techniques (Kudo and Matsumoto, 2001; Taskar et al., 2003). Tagging models are the best previous methods for text segmentation. However, their purely sequential form limits their ability to naturally handle overlapping or non-contiguous segments.

We present here an alternative view of segmentation as *structured multilabel classification*. In this view, a segmentation of a text is a set of *segments*, each of which is defined by the set of text positions that belong to the segment. Thus, a particular segment may not be a set of consecutive positions in the text, and segments may overlap. Given a text $\boldsymbol{x} = x_1 \cdots x_n$, the set of possible segments, which corresponds to the set of possible classification *labels*, is $\text{seg}(\boldsymbol{x}) = \{\texttt{O}, \texttt{I}\}^n$; for $\boldsymbol{y} \in \text{seg}(\boldsymbol{x})$, $y_i = \texttt{I}$ iff x_i belongs to the segment. Then, our segmentation task is to determine which labels are correct segments in a given text. We have thus a structured multilabel classification problem: each instance, a text, may have multiple structured labels, representing each of its segments. These labels are structured in that they do not come from a predefined set, but instead are built from sets of choices associated to the elements of arbitrarily long instances.

More generally, we may be interested in *typed* segments, e.g. segments naming different types of

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 987–994, Vancouver, October 2005.

entities. In that case, the set of segment labels is $\text{seg}(\boldsymbol{x}) = T \times \{\texttt{O}, \texttt{I}\}^n$, where T is the set of segment types. Since the extension is straightforward, we frame the discussion in terms of untyped segments, and only discuss segment types as needed.

At first sight, it might appear that we have made the segmentation problem intractably harder by turning it into a classification problem with a number of labels exponential on the length of the instance. However, we can bound the number of labels under consideration and take advantage of the structure of labels to find the k most likely labels efficiently. This will allow us to exploit recent advances in online discriminative methods for multilabel classification and ranking (Crammer and Singer, 2002).

Though multilabel classification has been well studied (Schapire and Singer, 1999; Elisseeff and Weston, 2001), as far as we are aware, this is the first study involving structured labels.

2 Segmentation as Tagging

The standard approach to text segmentation is to use tagging techniques with a BIO tag set. Elements in the input text are tagged with one of `B` for the beginning of a contiguous segment, `I` for the inside of a contiguous segment, or `O` for outside a segment. Thus, segments must be contiguous and non-overlapping. For instance, consider the sentence *Estimated volume was a light 2.4 million ounces*. Figure 1a shows how this sentence would be labeled using the BIO tag set for the problem of identifying base NPs in text. Given a particular tagging for a sentence, it is trivial to find all the segments, those whose tag sequences are longest matches for the regular expression $\texttt{BI}^*$. For typed segments, the BIO tag set is easily augmented to indicate not only segment boundaries, but also the type of each segment. Figure 1b exemplifies the tags for the task of finding people and organizations in text.

Sequential tagging with the BIO tag set has proven quite accurate for shallow parsing and named entity extraction tasks (Kudo and Matsumoto, 2001; Sha and Pereira, 2003; Tjong Kim Sang and De Meulder, 2003). However, this approach can only identify non-overlapping, contiguous segments. This is sufficient for some applications, and in any case, most training data sets are annotated without concern for overlapping or non-contiguous segments. However, there are instances in which sequential labeling techniques using the BIO label set will encounter problems.

Figure 2 shows two simple examples of segmentations involving overlapping, non-contiguous segments. In both cases, it is difficult to see how a sequential tagger could extract the segments correctly. It would be possible to grow the tag set to represent a bounded number of overlapping, non-contiguous segments by representing all possible combinations of segment membership over k overlapping segments, but this would require an arbitrary upper bound on k and would lead to models that generalize poorly and are expensive to train.

Dickinson and Meurers (2005) point out that, as language processing begins to tackle problems in free-word order languages and discourse analysis, annotating and extracting non-contiguous segmentations of text will become increasingly important. Though we focus primarily on entity extraction and NP chunking in this paper, there is no reason why ideas presented here could not be extended to managing other non-contiguous phenomena.

3 Structured Multilabel Classification

As outlined in Section 1, we represent segmentation as multilabel classification, assigning to each text the set of segments it contains. Figure 3 shows the segments for the examples of Figure 2. Each segment is given by a `O`/`I` assignment to its words, indicating which words belong to the segment.

By representing the segmentation problems as multilabel classification, we have fundamentally changed the objective of our learning and inference algorithms. The sequential tagging formulation is aimed to learn and find the best possible tagging of a text. In multilabel classification, we train model parameters so that correct labels — that is, correct segments – receive higher score than all incorrect ones. Likewise, inference becomes the problem of finding the set of correct labels for a text, that is, the set of correct segments.

We now describe the learning problem using the decision-theoretic multilabel classification and ranking framework of Crammer and Singer (2002) and Crammer (2005) as our starting point. In Sec-

```
a. Estimated volume was a light 2.4 million ounces .
      B        I     O  B   I    I     I      I    O

b. Bill   Clinton and Microsoft founder Bill   Gates  met today for 20 minutes .
   B-PER   I-PER   O    B-ORG      O    B-PER  I-PER   O    O    O  O     O    O
```

Figure 1: Sequential labeling formulation of text segmentation using the BIO label set. a) NP-chunking tasks. b) Named-entity extraction task.

```
a) Today, Bill and Hilary Clinton traveled to Canada.
   - Person: Bill Clinton
   - Person: Hilary Clinton
b)  ... purified bovine P450 11 beta / 18 / 19 - hydroxylase was ...
   - Enzyme: P450 11 beta-hydroxylase
   - Enzyme: P450 18-hydroxylase
   - Enzyme: P450 19-hydroxilase
```

Figure 2: Examples of overlapping and non-contiguous text segmentations.

tion 3.2, we describe a polynomial-time inference algorithm for finding up to k correct segments.

3.1 Training Multilabel Classifiers

Our model is based on a linear score $s(\boldsymbol{x}, \boldsymbol{y}; \mathbf{w})$ for each segment $\boldsymbol{y}$ of text $\boldsymbol{x}$, defined as

$$s(\boldsymbol{x}, \boldsymbol{y}; \mathbf{w}) = \mathbf{w} \cdot \mathbf{f}(\boldsymbol{x}, \boldsymbol{y})$$

where $\mathbf{f}(\boldsymbol{x}, \boldsymbol{y})$ is a feature vector representation of the sentence-segment pair, and $\mathbf{w}$ is a vector of feature weights. For a given text $\boldsymbol{x}$, $\text{act}(\boldsymbol{x}) \subseteq \text{seg}(\boldsymbol{x})$ denotes the set of correct segments for $\boldsymbol{x}$, and $\text{best}_k(\boldsymbol{x}; \mathbf{w})$ denotes the set of k segments with highest score relative to the weight vector $\mathbf{w}$. For learning, we use a training set $\mathcal{T} = \{(\boldsymbol{x}_t, \text{act}(\boldsymbol{x}_t))\}_{t=1}^{|\mathcal{T}|}$ of texts labeled with the correct segmentation.

We will discuss later the design of $\mathbf{f}(\boldsymbol{x}, \boldsymbol{y})$ and an efficient algorithm for finding the k highest scoring segments (where k is sufficiently large to include all correct segments). In this section, we present a method for learning a weight vector $\mathbf{w}$ that seeks to score correct segments above all incorrect segments.

Crammer and Singer (2002), extended by Crammer (2005), provide online learning algorithms for multilabel classification and ranking that take one instance at a time, construct a set of scoring constraints for the instance, and adjust the weight vector to satisfy the constraints. The constraints enforce a *margin* between the scores of correct labels and those of incorrect labels. The benefits of large-margin learning are best known from SVMs (Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2002), and are analyzed in detail by Crammer (2005) for online multilabel classification.

Training data: $\mathcal{T} = \{(\boldsymbol{x}_t, \text{act}(\boldsymbol{x}_t))\}_{t=1}^{|\mathcal{T}|}$
1. $\mathbf{w}^{(0)} = 0;\ i = 0$
2. for $n : 1..N$
3. for $t : 1..|\mathcal{T}|$
4. $\mathbf{w}^{(i+1)} = \arg\min_{\mathbf{w}} \left\|\mathbf{w} - \mathbf{w}^{(i)}\right\|^2$
   s.t. $s(\boldsymbol{x}_t, \boldsymbol{y}; \mathbf{w}) \geq s(\boldsymbol{x}_t, \boldsymbol{y}'; \mathbf{w}) + 1$
   $\forall \boldsymbol{y} \in \text{act}(\boldsymbol{x}_t), \forall \boldsymbol{y}' \in \text{best}_k(\boldsymbol{x}_t; \mathbf{w}^{(i)}) - \text{act}(\boldsymbol{x}_t)$
6. $i = i + 1$
7. $\mathbf{w} = \mathbf{w}^{(N*|\mathcal{T}|)}$

Figure 4: A simplified version of the multilabel learning algorithm of Crammer and Singer (2002).

For segmentation, the number of possible labels (segments) is exponential on the length of the text. We make the problem tractable by including only the margin constraints between correct segments and at most k highest scoring incorrect segments. Figure 4 sketches an online learning algorithm for multilabel classification based on the work of Crammer (2005). In the algorithm, $\mathbf{w}^{(i+1)}$ is the projection of $\mathbf{w}^{(i)}$ onto the set of weight vectors such that the scores of correct segments are separated by a margin of at least 1 from the scores of incorrect segments among the k top-scoring segments. This update is *conservative* in that there is no weight change if the constraint set is already satisfied or empty; if some constraints are not satisfied, we make the smallest weight change that satisfies the constraints. Since, the objective is quadratic in $\mathbf{w}$ and the constraints are linear, the optimization problem can be solved by Hildreth's al-

```
a) Today , Bill and Hilary Clinton traveled to Canada .
     O   O  I    O    O       I       O     O     O   O
     O   O  O    O    I       I       O     O     O   O

b)  ... purified bovine P450 11 beta / 18 / 19 - hydroxylase was ...
            O       O     I   I   I  O  O O  O I      I       O
            O       O     I   O   O  O  I O  O I      I       O
            O       O     I   O   O  O  O O  I I      I       O
```

Figure 3: Correct segments for two examples.

gorithm (Censor and Zenios, 1997).

Using standard arguments for linear classifiers (add constant feature, rescale weights) and the fact that all the correct scores in line 4 of Figure 4 are required to be above all the incorrect scores in the top k, that line can be replaced by

$$\begin{array}{l}\mathbf{w}^{(i+1)} = \arg\min_{\mathbf{w}} \left\|\mathbf{w} - \mathbf{w}^{(i)}\right\|^2 \\ \text{s.t. } s(\boldsymbol{x}_t, \boldsymbol{y}; \mathbf{w}) \geq 1 \text{ and } s(\boldsymbol{x}_t, \boldsymbol{y}'; \mathbf{w}) \leq -1 \\ \forall \boldsymbol{y} \in \text{act}(\boldsymbol{x}_t), \forall \boldsymbol{y}' \in \text{best}_k(\boldsymbol{x}_t; \mathbf{w}^{(i)}) - \text{act}(\boldsymbol{x}_t)\end{array}$$

If v is the number of correct segments for $\boldsymbol{x}$, this transformation replaces $O(kv)$ constraints with $O(k+v)$ constraints: segment scores are compared to a single positive or negative threshold rather then to each other. At test time, we find the segments with positive score by finding the k highest scoring segments and discarding those with a negative score.

3.2 Inference

During learning and at test time we require a method for finding the k highest scoring segments. At test time, we predict as correct all the segments with positive score in the top k. In this section we give an algorithm that calculates this precisely.

For inference, tagging models typically use the Viterbi algorithm (Rabiner, 1989). The algorithm is given by the following standard recurrences:

$$\begin{array}{rcl} S[i,t] & = & \max_{t'} s(t',t,i) + S[i-1,t'] \\ B[i,t] & = & \arg\max_{t'} s(t',t,i) + S[i-1,t'] \end{array}$$

with appropriate initial conditions, where $s(t',t,i)$ is the score for going from tag t' at $i-1$ to tag t at i. The dynamic programming table $S[i,t]$ stores the score of the best tag sequence ending at position i with tag t, and $B[i,t]$ is a back-pointer to the previous tag in the best sequence ending at i with t, which allows us to reconstruct the best sequence. The Viterbi algorithm has easy k-best extensions.

We could find the k highest scoring segments using Viterbi. However, for the case of non-contiguous segments, we would like to represent higher-order dependencies that are difficult to model in Viterbi. In particular, in Figure 3b we definitely want a feature bridging the gap between *Bill* and *Clinton*, which could not be captured with a standard first-order model. But moving to higher-order models would require adding dimensions to the dynamic programming tables S and B, with corresponding multipliers to the complexity of inference.

To represent dependencies between non-contiguous text positions, for any given segment $\boldsymbol{y} = y_1 \cdots y_n$, let $i(\boldsymbol{y}) = 0 i_1 \cdots i_m (n+1)$ be the increasing sequence of indices i_j such that $y_{i_j} = \texttt{I}$, padded for convenience with the dummy first index 0 and last index $n+1$. Also for convenience, set $x_0 = \texttt{-s-}$ and $x_{n+1} = \texttt{-e-}$ for fixed start and end markers. Then, we restrict ourselves to feature functions $\mathbf{f}(\boldsymbol{x}, \boldsymbol{y})$ that factor relative to the input as

$$\mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{|i(\boldsymbol{y})|} \mathbf{g}(i(\boldsymbol{y})_{j-1}, i(\boldsymbol{y})_j) \qquad (1)$$

where $i(\boldsymbol{y})_j$ is the j^{th} integer in $i(\boldsymbol{y})$ and $\mathbf{g}$ is a feature function depending on arbitrary properties of the input relative to the indices $i(\boldsymbol{y})_{j-1}$ and $i(\boldsymbol{y})_j$.

Applying (1) to the segment *Bill Clinton* in Figure 3, its score would be

$$\mathbf{w} \cdot [\mathbf{g}(0,3) + \mathbf{g}(3,6) + \mathbf{g}(6,11)]$$

This feature representation allows us to include dependencies between non-contiguous segment positions, as well as dependencies on any properties of the input, including properties of skipped positions.

We now define the following dynamic program

$$\begin{array}{rcl} S[i] & = & \max_{j<i} S[j] + \mathbf{w} \cdot \mathbf{g}(j,i) \\ B[i] & = & \arg\max_{j<i} S[j] + \mathbf{w} \cdot \mathbf{g}(j,i) \end{array}$$

These recurrences compute the score $S[i]$ of the best partial segment ending at i as the sum of the maximum score of a partial segment ending at position $j < i$, and the score of skipping from j to i. The back-pointer table B allows us to reconstruct the sequence of positions included in the segment.

Clearly, this program requires $O(n^2)$ time for a text of length n. Furthermore we can easily augment this algorithm in the standard fashion to find the k best segments, and multiple segment types, resulting in a runtime of $O(n^2kT)$, where T is the number of types. $O(n^2kT)$ is not ideal, but is still practical since in this work we are segmenting sentences. If we can bound the largest gap in any non-contiguous segment by a constant $g \ll n$, then the runtime can be improved to $O(ngkT)$. This runtime does not compare favorably to the standard Viterbi algorithm that runs in $O(nT^2)$, especially for large k. However, we found that for even large k we could still train large models in a matter of hours and test on unseen data in a few minutes.

3.2.1 Restrictions

Often a segmentation task or data set will restrict particular kinds of segments. For instance, it may be the case that a data set does not have any overlapping or non-contiguous segments. Embedded segmentations – those in which one segment's tokens are a subset of another's – is also a phenomenon that sometimes does not occur.

It is easy to restrict the inference algorithm to disallow such segments if they are unnecessary. For example, if two segments overlap or are embedded, the inference algorithm can just return the highest scoring one. Or it can simply ignore all non-contiguous segments if it is known that they do not occur in the data. In Section 4 we will augment the inference algorithm accordingly for each data set.

3.3 Feature Representation

We now discuss the design of the feature function for two consecutive segment positions $\mathbf{g}(j, i)$, where $j < i$. We build individual binary-valued features from predicates over the input, for instance, the identities of words in the sentence at particular positions relative to i and j. The selection of predicates varies by task, and we provide specific predicate sets in Section 4 for various data sets. In this section, we use for illustration word-pair identity predicates such as $x_j = \texttt{Bill} \,\&\, x_i = \texttt{Clinton}$.

For sequential tagging models, predicates are combined with the set of states (or tags) to create a feature representation. For our model, we define the following possible states:

$$\begin{aligned} \text{start} &\equiv j = 0 \\ \text{end} &\equiv i = n + 1 \\ \text{next} &\equiv j = i - 1 \\ \text{skip} &\equiv j < i - 1 \end{aligned}$$

For example, the following features would be on for $\mathbf{g}(0, 3)$[1] and $\mathbf{g}(3, 6)$, respectively, in Figure 3a:

$$x_j = \texttt{-s-} \,\&\, x_i = \texttt{Bill} \,\&\, \text{start}$$
$$x_j = \texttt{Bill} \,\&\, x_i = \texttt{Clinton} \,\&\, \text{skip}$$

These features indicate a predicate's role in the segment: at the beginning, at the end, over contiguous segment words or skipping over some words. All features can be augmented to indicate specific segment types for multi-type segmentation tasks. No matter what the task, we always add predicates that represent ranges of the distance $i-j$, as well as what words or part-of-speech tags occur between the two words. For instance, $\mathbf{g}(3, 6)$ might contain

$$\texttt{word-in-between} = \texttt{and} \,\&\, \text{skip}$$

These features are designed to identify common characteristics of non-contiguous segments such as the presence of conjunctions or punctuation in skipped portions. Although we have considered only binary features here, the model in principle allows arbitrary real-valued feature.

3.4 Summary

We presented a method for text segmentation that equates the problem to structured multilabel classification where each label corresponds to a segment. We showed that learning and inference can be managed tractably in the formulation by efficiently finding the k highest scoring segments through a dynamic programming algorithm that factors the structure of each segment. The only concern is that k must be large enough to include all correct segments,

[1] Note that "skip" is not on for $\mathbf{g}(0, 3)$ even though $j < i-1$. Start and end states override other states.

which we will discuss further in Section 4. This method naturally models all possible segmentations including those with overlapping or non-contiguous segments. Out approach can be seen as multilabel variant of the work of McDonald et al. (2004), which creates a set of constraints to separate the score of the single correct output from the k highest scoring outputs with an appropriate large margin.

4 Experiments

We now describe a set of experiments on named entity and base NP segmentation. For these experiments, we set $k = n$, where n is the length of the sentence. This represents a reasonable upper bound on the number of entities or chunks in a sentence and results in a time complexity of $O(n^3T)$.

We compare our methods with both the averaged perceptron (Collins, 2002) and conditional random fields (Lafferty et al., 2001) using identical predicate sets. Though all systems use identical predicates, the actual features of the systems are different due to the fundamental differences between the multilabel classification and sequential tagging models.

4.1 Standard data sets

Our first experiments are standard named entity and base NP data sets with no overlapping, embedded or non-contiguous segments. These experiments will show that, for simple segmentations, our model is competitive with sequential tagging models.

For the named entity experiments we used the CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003) data with people, organizations, locations and miscellaneous entities. We used standard predicates based on word, POS and orthographic information over a previous to next word window. For the NP chunking experiments we used the standard CoNLL 2000 data set (Kudo and Matsumoto, 2001; Sha and Pereira, 2003) using the predicate set defined by Sha and Pereira (2003).

The first three rows of Table 1 compare the multilabel classification approach to standard sequential classifiers. As one might expect, the performance of the multilabel classification method is below that of the sequential tagging methods. This is because those methods model contiguous segments well without the need for thresholds or k-best inference. In addition, the multilabel method shows significantly higher precision then recall. One possible reason for this is that during the course of learning, the model will see many segments that are *nearly* correct, e.g., segments that overlap correct segments and differ by a single token. As a result, the model learns to score all segments containing even a small amount of negative evidence as invalid in order to ensure that these nearly correct segments have a sufficiently low score.

One way to alleviate this problem is to restrict the inference algorithm to not return any overlapping, non-contiguous or embedded segmentations as discussed in Section 3.2.1, since this data set does not contain segments of this kind. This way, the learning stage only updates the parameters when a *nearly* correct segment actually out scores the correct one. The results of this system are shown in row 4 of Table 1. We can see that this change did lead to a more balanced precision/recall, however it is clear that more investigation is required.

4.2 Chemical substance extraction

The second set of experiments involves extracting chemical substance names from MEDLINE abstracts that relevant to the inhibition of the enzyme CYP450 (PennBioIE, 2005). We focus on abstracts that have at least one overlapping or non-contiguous annotation. This data set contains 6164 annotated chemical substances, including 6% that are both overlapping and non-contiguous. Figure 3b is an example from the corpus. We use identical predicates to the named entity experiments in Section 4.1. Though the data does contain overlapping and non-contiguous segments, it does not contain embedded segments. Results are shown in Table 2 using 10-fold cross validation. The sequential tagging models were trained using only sentences with no overlapping or non-contiguous entities. We found this provided the best performance. Row 4 of Table 2 shows the multilabel approach with the inference algorithm restricted to not allow embedded segments.

We can see that our method does significantly better on this data set (up to a 26% reduction in error). It is also apparent that the model is picking up some overlapping and non-contiguous entities (see Table 2). However, the models performance on these kinds of entities is lower than overall performance.

	a. **Named-Entity Extraction**			b. **NP-chunking**		
	Precision	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
Avg. Perceptron	82.46	83.14	82.80	94.22	93.88	94.05
CRFs	83.36	**83.57**	**83.47**	94.57	**94.00**	**94.29**
Multilabel	**92.47**	74.19	82.33	**94.65**	92.28	93.45
Multilabel with Restrictions	91.08	76.68	83.26	94.10	93.70	93.90

Table 1: Results for named-entity extraction and NP-chunking on data sets with only non-overlapping and contiguous segments annotated.

	Chem Substance Extraction - A			**Chem Substance Extraction - B**		
	Precision	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
Avg. Perceptron	82.98	79.40	81.15	**1.0**	0.0	0.0
CRFs	85.85	79.06	82.31	**1.0**	0.0	0.0
Multilabel	88.24	80.84	84.38	62.56	33.67	43.78
Multilabel with Restrictions	**88.55**	**84.59**	**86.53**	72.58	**45.92**	**56.25**

Table 2: Results for chemical substance extraction. Table A is for all entities in the data set and Table B is only for those entities that are overlapping and non-contiguous.

4.3 Tuning Precision and Recall

The learning algorithm in Section 3.1 seeks a separator through the origin, though, our experimental results suggest that this tends to favor precision at the expense of recall. However, at test time we can use a separation threshold different from zero. This parameter allows us to trade off precision against recall, and could be tuned on held-out data.

Figure 5 plots precision, recall and f-measure against the threshold for the basic multilabel model on the chemical substance, NP chunking and person entity extraction data sets. These plots clearly show what is expected: higher thresholds give higher precision, and lower thresholds give higher recall. In these data sets at least, a zero threshold is almost always near optimal, though sometimes we would benefit from a slightly lower threshold.

5 Discussion

We have presented a method for text segmentation that is base on discriminatively learning structured multilabel classifications. The benefits include

- Competitive performance with sequential tagging models.
- Flexible modeling of complex segmentations, including overlapping, embedded and non-contiguous segments.
- Adjustable precision-recall trade off.

However, there is a computation cost for our models. For a text of length n, training and testing require $O(n^3T)$ time, where T is the number of segment types. Fortunately, this still results in training times on the order of hours.

Our approach is related to the work of Bockhorst and Craven (2004). In this work, a conditional random field model is trained to allow for overlapping segments with an $O(n^2)$ inference algorithm. The model is applied to biological sequence modeling with promising results. However, our approaches differ in two major respects. First, their model is probabilistic, and trained to maximize segmentation likelihood, while our model is trained to maximize margin. Second, our method allows for non-contiguous segments, at the cost of a slower $O(n^3)$ inference algorithm.

In further work, the classification threshold should also be learned to achieve the desired balance between precision and recall. It would also be useful to investigate methods for combining these models with standard sequential tagging models to get top performance on simple segmentations as well as on overlapping or non-contiguous ones.

A broader area of investigation are other problems in language processing that can benefit from structured multilabel classification, e.g., ambiguities in language often result in multiple acceptable parses for sentences. It may be possible to extend the algorithms presented here to learn to distinguish all acceptable parses from unacceptable ones instead of just finding a single parse when many are valid.

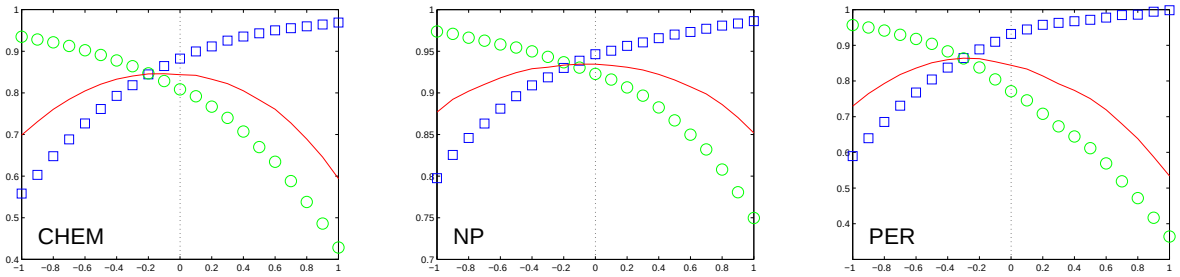

Figure 5: Precision (squares), Recall (circles) and F-measure (line) plotted against threshold values. CHEM: chemical substance extraction, NP: noun-phrase chunking, and PER: person name extraction.

Acknowledgments

We thank the members of the Penn BioIE project for the development of the CYP450 corpus that we used for our experiments. In particular, Seth Kulick answered many questions about the data. This work has been supported by the NSF ITR grant 0205448.

References

D.M. Bikel, R. Schwartz, and R.M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning Journal Special Issue on Natural Language Learning*, 34(1/3):221–231.

J. Bockhorst and M. Craven. 2004. Markov networks for detecting overlapping elements in sequence data. In *Proc. NIPS*.

Y. Censor and S.A. Zenios. 1997. *Parallel optimization : theory, algorithms, and applications*. Oxford University Press.

M. Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*.

K. Crammer and Y. Singer. 2002. A new family of online algorithms for category ranking. In *Proc SIGIR*.

K. Crammer. 2005. *Online Learning for Complex Categorial Problems*. Ph.D. thesis, Hebrew University of Jerusalem. to appear.

N. Cristianini and J. Shawe-Taylor. 2000. *An Introduction to Support Vector Machines*. Cambridge University Press.

M. Dickinson and W.D. Meurers. 2005. Detecting errors in discontinuous structural annotation. In *Proc. ACL*.

A. Elisseeff and J. Weston. 2001. A kernel method for multi-labeled classifi cation. In *Proc. NIPS*.

T. Kudo and Y. Matsumoto. 2001. Chunking with support vector machines. In *Proc. NAACL*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fi elds: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*.

A. McCallum, D. Freitag, and F. Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of ICML*.

R. McDonald, K. Crammer, and F. Pereira. 2004. Large margin online learning algorithms for scalable structured classication. In *NIPS Workshop on Structured Outputs*.

PennBioIE. 2005. Mining The Bibliome Project. http://bioie.ldc.upenn.edu/.

L. R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, February.

A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proc. EMNLP*.

R. E. Schapire and Y. Singer. 1999. Improved boosting algorithms using confi dence-rated predictions. *Machine Learning*, 37(3):1–40.

B. Schölkopf and A. J. Smola. 2002. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fi elds. In *Proc. HLT-NAACL*.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In *Proc. NIPS*.

E. F. Tjong Kim Sang and F. De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*. http://www.cnts.ua.ac.be/conll2003/ner.