Semi-Markov Models for Sequence Segmentation

Qinfeng Shi
NICTA, Statistical Machine Learning
Australian National University
Canberra, 2601 ACT
qinfeng.shi@rsise.anu.edu.au

Yasemin Altun
Toyota Technological Institute
1427 E 60th St
Chicago, IL 60637
altun@tti-c.org

Alex Smola
NICTA, Statistical Machine Learning
Australian National University
Canberra, 2601 ACT
Alex.Smola@nicta.com.au

S. V. N. Vishwanathan
NICTA, Statistical Machine Learning
Australian National University
Canberra, 2601 ACT
SVN.Vishwanathan@nicta.com.au

Abstract

In this paper, we study the problem of automatically segmenting written text into paragraphs. This is inherently a sequence labeling problem, however, previous approaches ignore this dependency. We propose a novel approach for automatic paragraph segmentation, namely training Semi-Markov models discriminatively using a Max-Margin method. This method allows us to model the sequential nature of the problem and to incorporate features of a whole paragraph, such as *paragraph coherence* which cannot be used in previous models. Experimental evaluation on four text corpora shows improvement over the previous state-of-the art method on this task.

1 Introduction

In this paper, we study automatic paragraph segmentation (APS). This task is closely related to some well known problems such as text segmentation, discourse parsing, topic shift detection and is relevant for various important applications in speech-to-text and text-to-text tasks.

In speech-to-text applications, the output of a speech recognition system, such as the output of systems creating memos and documents for the Parliament House, is usually raw text without any punctuation or paragraph breaks. Clearly, such text requires paragraph segmentations. In text-to-text processing, such as summarization, the output text does not necessarily retain the correct paragraph structure and may require post-processing. There is psycholinguistic evidence as cited by Sporleder & Lapata (2004) showing that insertion of paragraph breaks could improve the readability. Moreover, it has been shown that different languages may have cross-linguistic variations in paragraph boundary placement (Zhu, 1999), which indicates that machine translation can also benefit from APS. APS can also recover the paragraph breaks that are often lost in the OCR applications.

There has been growing interest within the NLP community for APS in recent years. Previous methods such as Sporleder & Lapata (2004); Genzel (2005); Filippova & Strube (2006) treat the problem as a binary classification task, where each sentence is labeled as the beginning of a paragraph or not. They focus on the use of features, such as surface features, language modeling features and syntactic features. The effectiveness of features is investigated across languages and/or domains. However, these approaches ignore the inherent sequential nature of APS. Clearly, consecutive sentences within the same paragraph depend on each other. Moreover, paragraphs should exhibit certain properties such as coherence, which should be explored within an APS system. One cannot incorporate such properties/features when APS is treated as a binary classification problem. To overcome this limitation, we cast APS as a sequence prediction problem, where the performance can be significantly improved by optimizing the choice of labeling over whole sequences of sentences, rather than individual sentences.

Sequence prediction is one of the most promi-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 640–648, Prague, June 2007.

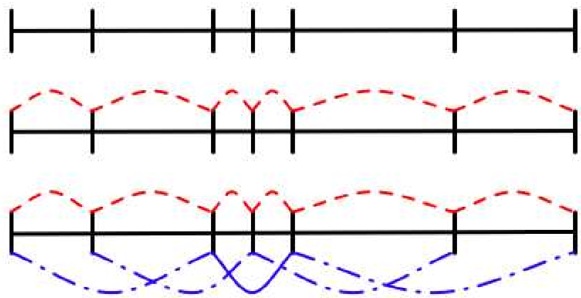

Figure 1: **Top**: sequence (horizontal line) with segment boundaries (vertical lines). This corresponds to a model where we estimate each segment boundary independently of all other boundaries. **Middle**: simple semi-Markov structure. The position of the segment boundaries only depends on the position of its neighbors, as denoted by the (red) dash arcs. **Bottom**: a more sophisticated semi-Markov structure, where each boundary depends on the position of two of its neighbors. This may occur, e.g., when the decision of where to place a boundary depends on the content of two adjacent segments. The longer range interaction is represented by the additional (blue) arcs.

nent examples of structured prediction. This problem is generally formalized such that there exists one variable for each observation in the sequence and the variables form a Markov chain (HMM). Segmentation of a sequence has been studied as a class of sequence prediction problems with common applications such as protein secondary structure prediction, Named Entity Recognition and segmentation of FAQ's. The exceptions to this approach are Sarawagi & Cohen (2004); Raetsch & Sonnenburg (2006), which show that Semi-Markov models (SMMs) (Janssen & Limnois, 1999), which are a variation of Markov models, are a natural formulation for sequence segmentation. The advantage of these models, depicted in Figure 1, is their ability to encode features that capture properties of a segment as a whole, which is not possible in an HMM model. In particular, these features can encode similarities between two sequence segments of arbitrary lengths, which can be very useful in tasks such as APS.

In this paper, we present a Semi-Markov model for APS and propose a max-margin training on these methods. This training method is a generalization of the Max-Margin methods for HMMs (Altun et al., 2003b) to SMMs. It follows the recent literature on discriminative learning of *structured prediction* (Lafferty et al., 2001; Collins, 2002; Altun et al., 2003a; Taskar et al., 2003). Our method inherits the advantages of discriminative techniques, namely the ability to encode arbitrary (overlapping) features and not making implausible conditional independence assumptions. It also has advantages of SMM models, namely the ability to encode features at segment level. We present a linear time inference algorithm for SMMs and outline the learning method. Experimental evaluation on datasets used previously on this task (Sporleder & Lapata, 2004) shows improvement over the state-of-the art methods on APS.

2 Modeling Sequence Segmentation

In sequence segmentation, our goal is to solve the estimation problem of finding a segmentation $y \in \mathcal{Y}$, given an observation sequence $x \in \mathcal{X}$. For example, in APS x can be a book which is a sequence of sentences. In a Semi-Markov model, there exists one variable for each subsequence of observations (i. e. multiple observations) and these variables form a Markov chain. This is opposed to an HMM where there exists one variable for each observation. More formally, in SMMs, $y \in \mathcal{Y}$ is a sequence of segment labelings $s_i = (b_i, l_i)$ where b_i is a non-negative integer denoting the beginning of the i^{th} segment which ends at position $b_{i+1} - 1$ and whose label is given by l_i (Sarawagi & Cohen, 2004). Since in APS the label of the segments is irrelevant, we represent each segment simply by the beginning position $y := \{b_i\}_{i=0}^{L-1}$ with the convention that $b_0 = 0$ and $b_L = N$ where N is the number of observations in x. Here, L denotes the number of segments in y. So the first segment is $[0, b_1)$, and the last segment is $[b_{L-1}, N)$, where $[a, b)$ denotes all the sentences from a to b inclusive a but exclusive b.

We cast this estimation problem as finding a discriminant function $F(x, y)$ such that for an observation sequence x we assign the segmentation that receives the best score with respect to F,

$$y^*(x) := \operatorname*{argmax}_{y \in \mathcal{Y}} F(x, y). \tag{1}$$

As in many learning methods, we consider functions that are linear in some feature representation Φ,

$$F(x,y;w) = \langle w, \Phi(x,y)\rangle. \tag{2}$$

Here, $\Phi(x,y)$ is a feature map defined over the joint input/output space as detailed in Section 2.3.

2.1 Max-Margin Training

We now present a maximum margin training for predicting structured output variables, of which sequence segmentation is an instance. One of the advantages of this method is its ability to incorporate the cost function that the classifier is evaluated with. Let $\Delta(y,\bar{y})$ be the cost of predicting $\bar{y}$ instead of y. For instance, Δ is usually the 0-1 loss for binary and multiclass classification. However, in segmentation, this may be a more sophisticated function such as the symmetric difference of y and $\bar{y}$ as discussed in Section 2.2. Then, one can argue that optimizing a loss function that incorporates this cost can lead to better generalization properties. One can find a theoretical analysis of this approach in Tsochantaridis et al. (2004).

We follow the general framework of Tsochantaridis et al. (2004) and look for a hyperplane that separates the correct labeling y_i of each observation sequence x_i in our training set from all the incorrect labelings $\mathcal{Y} - y_i$ with some margin that depends on Δ additively [1]. In order to allow some outliers, we use slack variables ξ_i and maximize the minimum margin, $F(x_i,y_i) - \max_{y\in\mathcal{Y}-y_i} F(x_i,y)$, across training instances i. Equivalently,

$$\min_{w,\xi}\ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \tag{3a}$$

$$\forall i,y\ \langle w, \Phi(x_i,y_i) - \Phi(x_i,y)\rangle \geq \Delta(y_i,y) - \xi_i. \tag{3b}$$

To solve this optimization problem efficiently, one can investigate its dual given by

$$\begin{aligned}\min_{\alpha}\ & \frac{1}{2}\sum_{i,j,y,y'} \alpha_{iy}\alpha_{jy'}\left\langle \Phi(x_i,y), \Phi(x_j,y')\right\rangle \\ & - \sum_{i,y}\Delta(y_i,y)\alpha_{iy} \\ \forall i,y\ & \sum_{y}\alpha_{iy} \leq C,\ \alpha_{iy}\geq 0.\end{aligned} \tag{4}$$

Here, there exists one parameter α_{iy} for each training instance x_i and its possible labeling $y \in \mathcal{Y}$. Solving this optimization problem presents a formidable challenge since $\mathcal{Y}$ generally scales exponentially with the number of variables within each variable y. This essentially makes it impossible to find an optimal solution via enumeration. Instead, one may use a column generation algorithm (Tsochantaridis et al., 2005) to find an approximate solution in polynomial time. The key idea is to find the most violated constraints (3b) for the current set of parameters and satisfy them up to some precision. In order to do this, one needs to find

$$\operatorname*{argmax}_{y\in\mathcal{Y}} \Delta(y_i,y) + \langle w, \Phi(x_i,y)\rangle\,, \tag{5}$$

which can usually be done via dynamic programming. As we shall see, this is an extension of the Viterbi algorithm for Semi Markov models.

Note that one can express the optimization and estimation problem in terms of kernels $k((x,y),(x',y')) := \langle \Phi(x,y), \Phi(x',y')\rangle$. We refer the reader to Tsochantaridis et al. (2005) for details.

To adapt the above framework to the segmentation setting, we need to address three issues: a) we need to specify a loss function Δ for segmentation, b) we need a suitable feature map Φ as defined in Section 2.3, and c) we need to find an algorithm to solve (5) efficiently. The max-margin training of SMMs was also presented in Raetsch & Sonnenburg (2006)

2.2 Cost Function

To measure the discrepancy between y and some alternative sequence segmentation y', we simply count the number of segment boundaries that have a) been missed and b) been wrongly added. Note that this definition allows for errors exceeding 100% - for

[1] There is an alternative formulation that is multiplicative in Δ. We prefer (3) due to computational efficiency reasons.

Algorithm 1 Max-Margin Training Algorithm

Input: data x_i, labels y_i, sample size m, tolerance ϵ

Initialize $S_i = \emptyset$ for all i, and $w = 0$.

repeat

  for $i = 1$ **to** m **do**

    $w = \sum_i \sum_{y \in S_i} \alpha_{iy} \Phi(x_i, y)$

    $y^* = \text{argmax}_{y \in \mathcal{Y}} \langle w, \Phi(x_i, y) \rangle + \Delta(y_i, y)$

    $\xi = \max(0, \max_{y \in S_i} \langle w, \Phi(x_i, y) \rangle + \Delta(y_i, y))$

    if $\langle w, \Phi(x_i, y^*) \rangle + \Delta(y_i, y) > \xi + \epsilon$ **then**

      Increase constraint set $S_i \leftarrow S_i \cup y^*$

      Optimize (4) wrt $\alpha_{iy}, \forall y \in S_i$.

    end if

  end for

until S has not changed in this iteration

instance, if we were to place considerably more boundaries than can actually be found in a sequence.

The number of errors is given by the symmetric difference between y and y', when segmentations are viewed as sets. This can be written as

$$\begin{aligned} \Delta(y, y') &= |y| + |y'| - 2|y \cap y'| \\ &= |y| + \sum_{i=1}^{l'} \left[1 - 2\left\{b'_i \in y\right\}\right]. \end{aligned} \tag{6}$$

Here $|\cdot|$ denotes the cardinality of the set. Eq. (6) plays a vital role in solving (5), since it allows us to decompose the loss in y' into a constant and functions depending on the segment boundaries b'_i only. Note that in the case where we want to segment *and* label, we simply would need to check that the positions are accurate *and* that the labels of the segments match.

2.3 Feature Representation

SMMs can extract three kinds of features from the input/output pairs: a) node features, i. e. features that encode interactions between attributes of the observation sequence and the (label of a) *segment* (rather than the label of each observation as in HMM), b) features that encode interactions between neighboring labels along the sequence and c) edge features, i. e. features that encode properties of segments. The first two types of features are commonly used in other sequence models, such as HMMs and Conditional Random Fields (CRFs). The third feature type is specific to Semi-Markov models. In particular, these features can encode properties of a whole segment or similarities between two sequence segments of arbitrary lengths. The cost of this expressibility is simply a constant factor of the complexity of Markov models, if the maximum length of a segment is bounded. This type of features are particularly useful in the face of sparse data.

As in HMMs, we assume stationarity in our model and sum over the features of each segment to get $\Phi(x, y)$. Then, Φ corresponding to models of the middle structure given in Figure 1 is given by

$$\Phi(x, \bar{y}) := (\Phi_0, \sum_{i=1}^{\bar{l}-1} \Phi_1(\bar{n}_i, x), \sum_{i=1}^{\bar{l}} \Phi_2(\bar{b}_{i-1}, \bar{b}_i, x)).$$

We let $\Phi_0 = \bar{l} - 1$, the number of segments. The node features Φ_1 capture the dependency of the current segment boundary to the observations, whereas the edge features Φ_2 represent the dependency of the current segment to the observations. To model the bottom structure in Figure 1, one can design features that represent the dependency of the current segment to its adjacent segments as well as the observations, $\Phi_3(x, b_{i-2}, b_{i-1}, b_i)$. The specific choices of the feature map Φ are presented in Section 3.

2.4 Column Generation on SMMs

Tractability of Algorithm 1 depends on the existence of an efficient algorithm that finds the most violated constraint (3b) via (5). Both the cost function of Section 2.2 and the feature representation of Section 2.3 are defined over a short sequence of segment boundaries. Therefore, using the Markovian property, one can perform the above maximization step efficiently via a dynamic programming algorithm. This is a simple extension of the Viterbi algorithm. The inference given by (1) can be performed using the same algorithm, setting Δ to a constant function.

We first state the dynamic programming recursion for $F + \Delta$ in its generality. We then give the pseudocode for $\Phi_3 = \emptyset$.

Denote by $T(t_-, t_+; x)$ the largest value of $\Delta(y, p) + F(x, p)$ for any *partial* segmentation p that starts at position 0 and which ends with the segment $[t_-, t_+)$. Moreover, let M be a upper bound on the

Algorithm 2 Column Generation

Input: sequence x, segmentation y, max-length of a segment M
Output: score s, segment boundaries y'
Initialize vectors $T \in \mathbb{R}^m$ and $R \in \mathcal{Y}^m$ to 0
for $i = 1$ **to** l **do**
  $R_i = \operatorname*{argmax}_{\max(0,i-M)\leq j<i} T_j + g(j,i)$
  $T_i = T_{R_i} + g(R_i, i)$
end for
$s = T_m + |y|$
$y' = \{m\}$
repeat
  $i = y'_{\text{first}}$
  $y' \leftarrow \{R_i, y'\}$
until $i = 0$

length of a segment. The recursive step of the dynamic program is given by

$$\begin{aligned} T(t_-, t_+; x) &= \max_{\max(0,t_- - M)\leq k<t_-} T(k, t_-; x) \\ &+ g(k, t_-, t_+) \end{aligned}$$

where we defined the increment $g(k, t_-, t_+)$ as

$$\langle \Phi_0(x), \Phi_1(x, t_+), \Phi_2(x, t_-, t_+), \Phi_3(x, k, t_-, t_+), w\rangle \\ + 1 - 2\{(t_-, t_+) \in y\}$$

where by convention $T(i, i') = -\infty$ if $i < 0$ for all labels. Since T needs to be computed for all values of $t_+ - M \leq t_- < t_+$, we need to compute $O(|x|M)$ many values, each of which requires an optimization over M possible values. That is, storage requirements are $O(|x|M)$, whereas the computation scales with $O(|x|M^2)$. If we have a good bound on the maximal sequence length, this can be dealt with efficiently. Finally, the recursion is set up by $T(0, 0, x) = |y|$.

See Algorithm 2 for pseudocode, when $\Phi_3 = \emptyset$. The segmentation corresponding to (5) is found by constructing the path traversed by the argument of the $\max$ operation generating T.

3 Features

We now specify the features described in Section 2.3 for APS. Note that the second type of features do not exist for APS since we ignore the labelings of segments.

3.1 Node Features Φ_1

Node features $\Phi_1(b_j, x)$ represent the information of the current segment boundary and some attributes of the observations around it (which we define as the current, preceding and successive sentences). These are sentence level features, which we adapt from Genzel (2005) and Sporleder & Lapata (2004) [2]. For the b_jth sentence, $x(b_j)$, we use the following features

- Length of $x(b_j)$.
- Relative Position of $x(b_j)$.
- Final punctuation of $x(b_j)$.
- Number of capitalized words in $x(b_j)$.
- Word Overlap of $x(b_j)$ with the next one

$$W_{over}(x(b_j), x(b_j + 1)) = \\ \frac{2 \mid x(b_j) \cap x(b_j + 1) \mid}{\mid x(b_j) \mid + \mid x(b_j + 1) \mid}.$$

- First word of $x(b_j)$.
- Bag Of Words (BOW) features: Let the bag of words of a set of sentences S be

$$B(S) = (c_0, c_1, ..., c_i, ..., c_{N-1}),$$

  where N is the size of the dictionary and c_i is the frequency of word i in S.
  - BOW of $x(b_j)$, $B(\{x(b_j)\})$
  - BOW of $x(b_j)$ and the previous sentence $B(\{x(b_j - 1), x(b_j)\})$
  - BOW of $x(b_j)$ and the succeeding sentence $B(\{x(b_j), x(b_j + 1)\})$
  - The inner product of the two items above
- Cosine Similarity of $x(b_j)$ and the previous sentence

$$CS(x(b_j - 1), x(b_j)) \\ = \frac{\langle B(x(b_j - 1)), B(x(b_j))\rangle}{\mid B(x(b_j - 1)) \mid \times \mid B(x(b_j)) \mid}$$

[2]Due to space limitations, we omit the motivations for these features and refer the reader to the literature cited above.

- Shannon's Entropy of $x(b_j)$ computed by using a language model as described in Genzel & Charniak (2003).

- Quotes(Q_p, Q_c, Q_i). Q_p and Q_c are the number of pairs of quotes in the previous(Num_p) and current sentence (Num_c), $Q_p = 0.5 \times Num_p$ and $Q_c = 0.5 \times Num_c$.

3.1.1 Edge Features Φ_2

Below is the set of features $\Phi_2(b_j, b_{j+1}, x)$ encoding information about the current segment. These features represent the power of the Semi-Markov models. Note that Φ_3 features also belong to edge features category. In this paper, we did not use Φ_3 feature due to computational issues.

- Length of The Paragraph: This feature expresses the assumption that one would want to have a balance across the lengths of the paragraphs assigned to a text. Very long and very short paragraphs should be uncommon.

- Cosine Similarity of the current paragraph and neighboring sentences: Ideally, one would like to measure the similarity of two consecutive paragraphs and search for a segmentation that assigns low similarity scores (in order to facilitate changes in the content). This can be encoded using $\Phi_3(x, b_{j-1}, b_j, b_{j+1})$ features. When such features are computationally expensive, one can measure the similarity of the current paragraph with the preceding sentence as

$$CS(P, x(b_j - 1)) = \frac{\langle BOW(P), BOW(x(b_j - 1)) \rangle}{\mid BOW(P) \mid \times \mid BOW(x(b_j - 1)) \mid}$$

  where P is the set of sentences in the current paragraph, $[b_j, b_{j+1})$. A similar feature is used for $CS(P, x(b_{j+1}))$.

- Shannon's Entropy of the Paragraph: The motivation for including features encoding the entropy of the sentences is the observation that the entropy of paragraph initial sentences is lower than the others (Genzel & Charniak, 2003). The motivation for including features encoding the entropy of the paragraphs, on the other hand, is that the entropy rate should remain more or less constant across paragraphs, especially for long texts like books. We ignore the sentence boundaries and use the same technique that we use to compute the entropy of a sentence.

3.2 Feature Rescaling

Most of the features described above are binary. There are also some features such as the entropy whose value could be very large. We rescale all the non-binary valued features so that they do not override the effect of the binary features. The scaling is performed as follows:

$$u_{new} = \frac{u - \min(u)}{\max(u) - \min(u)}$$

where u_{new} is the new feature and u is the old feature. $\min(u)$ is the minimum of u, and $\max(u)$ is the maximum of u. An exception to this is the rescaling of BOW features which is given by

$$B(x(b_j))_{new} = B(x(b_j)) / \langle B(x(b_j)), B(x(b_j)) \rangle$$

$\langle ., . \rangle$ denotes the inner product.

4 Experiments

We collected four sets of data for our experiments. The first corpus, which we call SB, consists of manually annotated text from the book *The Adventures of Bruce-Partington Plans* by Arthur Conan-Doyle. The second corpus, which we call SA, again consists of manually annotated text but from 10 different books by Conan-Doyle. Our third corpus consists of German (GER) and English (ENG) texts. The German data consisting of 12 German novels was used by Sporleder & Lapata (2006). This data uses automatically assigned paragraph boundaries, with the labeling error expected to be around 10%. The English data contains 12 well known English books from Project Gutenberg (`http://www.gutenberg.org/wiki/Main_Page`). For this dataset the paragraph boundaries were marked manually.

All corpora were approximately split into training (72%), development (21%), and test set (7%) (see Table 1). The table also reports the accuracy of the baseline classifier, denoted as BASE, which either labels all sentences as paragraph boundaries or

Table 1: Number of sentences and % accuracy of the baseline classifier (BASE) on various datasets used in our experiments.

	TOTAL	TRAIN	DEV	TEST	BASE
SB	59,870	43,678	12,174	3,839	53.70
SA	69,369	50,680	14,204	4,485	58.62
ENG	123,261	88,808	25,864	8,589	63.41
GER	370,990	340,416	98,610	31,964	62.10

Table 2: Test results on ENG and GER data after model selection.

DATASET	ALGO.	ACC.	REC.	PREC.	F_1
ENG	SMM	**75.61**	**46.67**	**77.78**	**58.33**
	SVM	58.54	26.67	40.00	32.00
	BT	65.85	33.33	55.56	41.67
GER	SMM	70.56	46.81	65.67	54.66
	SVM	39.92	**100.00**	38.68	55.79
	BT	**72.58**	54.26	**67.11**	**60.00**

non-boundaries, choosing whichever scheme yields a better accuracy.

We evaluate our system using accuracy, precision, recall, and the F_1-score given by $(2 \times Precision \times Recall)/(Precision+Recall)$ and compare our results to Sporleder & Lapata (2006) who used BoosTexter (Schapire & Singer, 2000) as a learning algorithm. To the best of our knowledge, BoosTexter (henceforth called BT) is the leading method published for this task so far. In order to evaluate the importance of the edge features and the resultant large-margin constraint, we also compare against a standard binary Support Vector Machine (SVM) which uses node features alone to predict whether each sentence is the beginning of a paragraph or not. For a fair comparison, all classifiers used the linear kernel and the same set of node features.

We perform model selection for all three algorithms by choosing the parameter values that achieve the best F_1-score on the development set. For both the SVM as well as our algorithm, SMM, we tune the parameter C (see (3a)) which measures the trade-off between training error and margin. For BT, we tune the number of Boosting iterations, denoted by N.

4.1 Results

In our first experiment, we compare the performance of our algorithm, SMM, on the English and German corpus to a standard SVM and BoosTexter. We report these result in Table 2. Our algorithm achieves the best F_1-score on the ENG corpus. SMM performs very competitively on the GER corpus, achieving accuracies close to those of BT.

We observed a large discrepancy between the performance of our algorithm on the development and the test datasets. The situation is similar for both SVM and BT. For instance, BT when trained on the ENG corpora, achieves an optimal F_1-score of 18.67% after $N = 100$ iterations. For the same N value, the test performance is 41.67%. We conjecture that this discrepancy is because the books that we use for training and test are written by different authors. While there is some generic information about when to insert a paragraph break, it is often subjective and part of the authors style. To test this hypothesis, we performed experiments on the SA and SB corpus, and present results in Table 3. Indeed, the F_1-scores obtained on the development and test corpus closely match for text drawn from the same book (whilst exhibiting better overall performance), differs slightly for text drawn from different books by the same author, and has a large deviation for the GER and ENG corpus.

Table 3: Comparison on various ENG datasets.

DATASET	ACC.	REC.	PREC.	F_1-SCORE
SB (DEV)	92.81	86.44	92.73	89.47
SB (TEST)	96.30	96.00	96.00	96.00
SA (DEV)	82.24	61.11	82.38	70.17
SA (TEST)	81.03	79.17	76.00	77.55
ENG (DEV)	69.84	18.46	78.63	29.90
ENG (TEST)	75.61	46.67	77.78	58.33

There is one extra degree of freedom that we can optimize in our model, namely the offset, i. e. the weight assigned to the constant feature Φ_0. After fixing all the parameters as described above, we vary the value of the offset parameter and pick the value that gives the F_1-score on the development data. We choose to use F_1-score, since it is the error measure that we care about. Although this extra optimization leads to better F_1-score in German (69.35% as opposed to 54.66% where there is no extra tuning of the offset), it results in a decrease of the F_1-score in English (52.28% as opposed to 58.33%). These results are reported in Table 4. We found that the difference of the F_1-score of tuning and not tuning the threshold on the development set was not a good indicator on the usefulness of this extra parameter. We are now investigating other properties, such as variance on the development data, to see if the tuning of the threshold can be used for better APS systems.

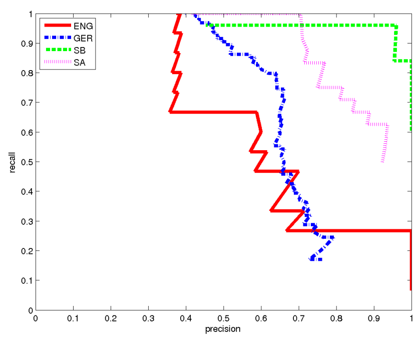

Figure 2: Precision-recall curves

Figure 2 plots the precision-recall curve obtained on various datasets. As can be seen the performance of our algorithm on the SB dataset is close to optimum, whilst it degrades slightly on the SA dataset, and substantially on the ENG and GER datasets. This further confirms our hypothesis that our algorithm excels in capturing stylistic elements from a single author, but suffers slightly when trained to identify generic stylistic elements. We note that this is not a weakness of our approach alone. In fact, all the other learning algorithms also suffer from this shortcoming.

Table 4: Performance on ENG test set tuning the offset for best F_1-score on ENG development set.

DATASET	ACC.	REC.	PREC.	F_1-SCORE
ENG	75.61	46.67	77.78	58.33
ENG $+\Phi_0$	39.02	93.33	36.84	52.28
GER	70.56	46.81	65.67	54.66
GER $+ \Phi_0$	75.40	73.40	65.71	69.35

5 Conclusion

We presented a competitive algorithm for paragraph segmentation which uses the ideas from large margin classifiers and graphical models to extend the semi-Markov formalism to the large margin case. We obtain an efficient dynamic programming formulation for segmentation which works in linear time in the length of the sequence. Experimental evaluation shows that our algorithm is competitive when compared to the state-of-the-art methods.

As future work, we plan on implementing Φ_3 features in order to perform an accuracy/time analysis. By defining appropriate features, we can use our method immediately for text and discourse segmentation. It would be interesting to compare this method to Latent Semantic Analysis approaches for text segmentation as studied for example in Bestgen (2006) and the references thereof.

References

Altun, Y., Hofmann, T., & Johnson, M. (2003a). Discriminative Learning for Label Sequences via Boosting. In *In Proceedings of NIPS 2003*.

Altun, Y., Tsochantaridis, I., & Hofmann, T. (2003b). Hidden markov support vector machines. In *International Conference on Machine Learning*.

Bestgen, Y. (2006). Improving text segmentation using latent semantic analysis: A reanalysis of choi, wiemer-hastings, and moore (2001). *Computational Linguistics*, *32*, 5–12.

Collins, M. (2002). Discriminative training methods for hidden markov models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Filippova, K., & Strube, M. (2006). Using linguistically motivated features for paragraph segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Genzel, D. (2005). A paragraph boundary detection system. In *Proceedings of the Conference on Computational Linguistics and Intelligent Text Processing*.

Genzel, D., & Charniak, E. (2003). Variation of entropy and parse tree of sentences as a function of the sentence number. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Janssen, J., & Limnois, N. (1999). *Semi-markov models and applications*. Kluwer Academic.

Lafferty, J. D., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic modeling for segmenting and labeling sequence data. In *18th International Conference on Machine Learning ICML*.

Raetsch, G., & Sonnenburg, S. (2006). Large scale hidden Semi-Markov SVMs for gene structure prediction. In *In Proceedings of NIPS 2006*.

Sarawagi, S., & Cohen, W. (2004). Semi-Markov Conditional Random Fields for Information Extraction. In *Advances in Neural Information Processing Systems (NIPS)*.

Schapire, R. E., & Singer, Y. (2000). Boostexter: A boosting-based system for text categorization. *Machine Learning*, *39*(2/3), 135–168.

Sporleder, C., & Lapata, M. (2004). Automatic paragraph identification: A study across languages and domains. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Sporleder, C., & Lapata, M. (2006). Broad coverage paragraph segmentation across languages and domains. *ACM Trans. Speech Lang. Process.*, *3*(2), 1–35.

Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin markov networks. In S. Thrun, L. Saul, & B. Schölkopf, eds., *Advances in Neural Information Processing Systems 16*.

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *ICML '04: Twenty-first international conference on Machine learning*. New York, NY, USA: ACM Press. ISBN 1-58113-828-5.

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*.

Zhu, C. (1999). Ut once more: The sentence as the key functional unit of translation. *Meta*, *44(3)*, 429–447.