Attention-Based Neural Text Segmentation

Pinkesh Badjatiya(✉), Litton J. Kurisinkel, Manish Gupta, and Vasudeva Varma

IIIT-H, Hyderabad, India
{pinkesh.badjatiya,litton.jkurisinkel}@research.iiit.ac.in,
{manish.gupta,vv}@iiit.ac.in

Abstract. Text segmentation plays an important role in various Natural Language Processing (NLP) tasks like summarization, context understanding, document indexing and document noise removal. Previous methods for this task require manual feature engineering, huge memory requirements and large execution times. To the best of our knowledge, this paper is the first one to present a novel supervised neural approach for text segmentation. Specifically, we propose an attention-based bidirectional LSTM model where sentence embeddings are learned using CNNs and the segments are predicted based on contextual information. This model can automatically handle variable sized context information. Compared to the existing competitive baselines, the proposed model shows a performance improvement of ∼7% in WinDiff score on three benchmark datasets.

1 Introduction

The task of text segmentation is defined as the process of segmenting a chunk of text into meaningful sections based on their topical continuity. Text segmentation is one of the fundamental NLP problems which finds its use in many tasks like summarization [14], passage extraction [16], discourse analysis [26], Question-Answering [16], context understanding, document noise removal, etc. Fine grained segmentation of a document into multiple sections provides a better understanding about the document structure which can also be used to generate better document representations, which in turn could benefit other natural language applications. Complexity of text segmentation varies with the type of text and writing styles – informational, conversational, narrative, descriptive, etc. In some cases, context is a very important signal for the task, while in other cases, dependence on context may be minimal. Also, complex topic shifts in the text and use of abstract cue phrases in the sentences make the task challenging.

Multiple supervised and unsupervised methods have been already proposed to tackle some of these challenges. Many unsupervised methods are heuristic and ad hoc in nature, need huge memory, have large execution times, and

Manish Gupta—A Principal Applied Scientist at Microsoft.

do not generalize well across multiple text types. Supervised methods require labeled data and often the performance of such systems comes at the cost of hand-crafted highly tuned feature engineering. None of these methods can automatically tune the degree of dependence on the context. Sequence-to-sequence models like Recurrent Neural Networks (RNNs) and Long Term-Short Memory (LSTMs) can model sequences effectively by controlling information flow across time. Such models can in general help capture long range dependencies (context) but they work well with short sequences. They can be enhanced by giving varying attention weights to sentences in the context, where the weight denotes the relative importance of a context sentence for segmentation. Attention thus allows us to learn the focus points from the context. To the best of our knowledge, this is the first work to explore the use of attention-based neural mechanism for text segmentation. We propose a novel Attention-based CNN-BiLSTM model that learns to represent the context of the sentence by learning the attention weights. The proposed model does not require any manually designed features, is domain independent and scalable. The proposed neural model architecture is illustrated in Fig. 1. We compare the proposed method with competitive baselines on three benchmark datasets.

In Sect. 2, we review the existing work on text segmentation. Section 3 describes the proposed Neural model with Attention-based approach. Section 4 compares the performance of various methods on benchmark datasets. In Sect. 5, we analyze the results and conclude with a brief summary in Sect. 6.

2 Related Work

Unsupervised methods for text segmentation include lexical cohesion [4,8], statistical modeling [1,25], affinity propagation based clustering [11,22], and topic modeling [4,13,18,20]. Topic modeling approaches include PLDA [18] (captures the amount of topic distribution that a paragraph shares with its predecessor), SITS [15] (chains a set of Hierarchical Dirichlet Process LDAs), TSM [2] (integrates point-wise boundary sampling with topic modeling), and [3] (ordering-based probabilistic topic models to incorporate the ordering irregularity into the probabilistic approach). These methods are globally informed, i.e., they consider the whole document when generating the most probable segment boundaries. However, huge memory requirements and large execution times make these methods unpractical for use in real applications.

Various classifiers like decision trees [6,24] and probabilistic models [1,7, 19,25] have been proposed for supervised text segmentation. Popular features include lexical (like lexical similarities [8]), conversational (acoustic indicators, long pauses, shifts in speaking rates, higher maximum accent peak, cue phrases, silences, overlaps, speaker change [5]) and knowledge-based features [10]. Supervised methods require labeled data and hand-crafted highly tuned feature engineering. Also, they are locally informed and often fail to capture the overall global topic structure of the document.

Some previous studies, although scarce and somewhat preliminary, have explored neural approaches for domain-specific text segmentation. Sheikh et al. [23] proposed a method for segmentation in transcripts using RNNs, Wang et al. [28] attempt to learn a coherence function using the partial ordering relations, Wang et al. [27] use BiLSTM-CNN to model the task as a simple binary classification task for Chinese. In this paper we explore the use of attention-based deep neural architecture for the task of automatic linear text segmentation, which provides a good trade off between the locally informed and globally informed behavior by varying the amount of context information used.

3 The Proposed Method

In this section, we start by presenting the formal problem definition. Further we discuss steps related to the data preparation and pre-processing. Finally, we present our neural model architecture.

3.1 Problem Definition

We model the text segmentation problem as a binary classification problem. Given a document, we define the problem with respect to the i^{th} sentence in the document, as follows.

Given: A sentence s_i with its K sized *left-context* $\{s_{i-K}, \ldots, s_{i-1}\}$ (i.e., K sentences before s_i) and K sized *right-context* $\{s_{i+1}, \ldots, s_{i+K}\}$ (i.e., K sentences after s_i). Here K is the context size.
Predict: Whether the sentence s_i denotes the beginning of a new text segment.

In this paper, we propose a neural framework to tackle this problem. Using a neural framework, we aim at using the context for learning distinctive features for sentences that mark the beginning of the segment. The architecture of the proposed model is illustrated in Fig. 1.

3.2 Data Preparation

In this section, we discuss two main steps in data preparation: pre-processing and custom batch creation to incorporate neighboring context.

Data Pre-processing. We fix the length of sentences to L words and truncate/pad as required to achieve appropriate fixed length embedding of the sentences. To represent words, we use the 300D word2vec[1] embeddings which are trained on Google News dataset containing ∼100B words with a vocabulary size of ∼3M words.

Let V represent the vocabulary, and let d be the word embedding size. Let $E^{V \times d}$ be the embedding matrix whose each row represents the embedding of

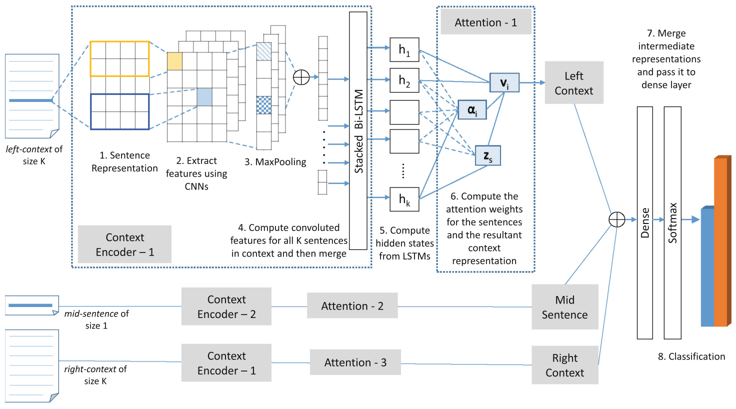
[1] https://code.google.com/archive/p/word2vec/.

a particular word in the model vocabulary. Let η_i be a matrix whose j^{th} row corresponds to the one-hot representation of the j^{th} word of the sentence s_i. Thus, η_i has L rows and V columns. Given the word embedding matrix E, we obtain the sentence embedding matrix $e(s_i)$ for sentence s_i as $e(s_i)^{L \times d} = \eta_i^{L \times V} \times E^{V \times d}$.

While creating η_i for a sentence s_i, we perform basic text cleaning steps like skipping the punctuations and stop words. For all the missing words in the word2vec vocabulary we perform WordNet-based lemmatization and use the lemmatized word instead. If the embedding is still missing from the vocabulary then we replace it with a special token ⟨UNK⟩.

Custom Batch Creation. We wish to exploit the context around a sentence to decide whether the sentence indicates a segment boundary. For a sentence s_i, let lc_i and rc_i be the one-hot representations of the *left-context* and *right-context* respectively. Both lc_i and rc_i therefore contain $K \times L$ words each. Their embeddings $e(lc_i)$ and $e(rc_i)$ can then be computed as $e(lc_i)^{K \times L \times d} = lc_i^{K \times L \times V} \times E^{V \times d}$ and $e(rc_i)^{K \times L \times d} = rc_i^{K \times L \times V} \times E^{V \times d}$ respectively. Note that we also refer to the sentence s_i as the *mid-sentence*. We perform padding as required to obtain a fixed length representation of size $K \times L \times d$ for both the contexts. The input to the model is a batch of samples with the i^{th} sample, S_i, defined as the concatenation of the embeddings of the *left-context*, *mid-sentence* and the *right-context* as follows.

$$S_i = [e(lc_i)^{K \times L \times d}, e(s_i)^{L \times d}, e(rc_i)^{K \times L \times d}] \tag{1}$$



left-context of size K
1. Sentence Representation
2. Extract features using CNNs
3. MaxPooling
4. Compute convoluted features for all K sentences in context and then merge
Context Encoder – 1
Stacked Bi-LSTM
h1
h2
hk
5. Compute hidden states from LSTMs
Attention - 1
vi
αi
zs
6. Compute the attention weights for the sentences and the resultant context representation
Left Context
7. Merge intermediate representations and pass it to dense layer
Dense
Softmax
8. Classification
mid-sentence of size 1
Context Encoder – 2
Attention - 2
Mid Sentence
right-context of size K
Context Encoder – 1
Attention - 3
Right Context

Fig. 1. Architecture diagram for the proposed model

The context size K should be such that the covered neighborhood information is enough to make conclusive decision about the current sentence being a segment boundary or not. Higher values of K provide the model with extra unnecessary context along with increase in the number of parameters. Lower values of K reduces the model complexity, but also restricts the model's ability to capture relations across near sentences only. K can be tuned using validation data. We study sensitivity of results to variation in K in Sect. 5.

3.3 The Proposed Neural Model

We now discuss in detail the proposed neural network model as illustrated in Fig. 1. The data pre-processing and sentence embedding discussed in the previous sub-section provides us instances of the form S_i, which consist of *left-context*, *right-context* and the *mid-sentence* word-embedding representations. This corresponds to the output at Step 1 in Fig. 1.

CNN Transformations. We leverage the widely used CNN architecture to obtain rich feature representations for each sentence in the *left-context*, *mid-sentence* as well as the *right-context*. Recall that the embedding S_i has $2K + 1$ rows each having $L \times d$ dimensions. Let us denote the j^{th} such embedding matrix in S_i as $S_{(i,j)}^{L \times d}$. On each such $S_{(i,j)}$, we perform 1D Convolution operations with z number of filters. Let us denote the l^{th} filter with a set of weights $\{\omega_l^{h \times d}, b_l\}$. Such a filter with height h can be applied on the input $S_{(i,j)}$ to obtain feature maps as follows.

$$f_{kl} = \phi(\omega_l^{h \times d} \cdot S_{(i,j)}[k - \frac{h}{2} : k + \frac{h}{2}]^{h \times d} + b_l) \tag{2}$$

Note that the convolution operations on text data involve filters with width same as input dimensionality (d). Thus, a filter has dimensions $d \times h$. Here, f_{kl} denotes the result of the convolution using a non-linear transformation ϕ. The filter is applied to each row of $S_{i,j}$. After applying z such filters, for each row k of $S_{i,j}$, we obtain a feature vector $f_k = \{f_{k1}, f_{k2}, \ldots, f_{kz}\}$. This corresponds to the output at Step 2 of Fig. 1.

Max-pooling is a popular sample-based discretization operation in CNNs. Given the feature vector f_k, max pooling operation involves computing the maximum feature value per filter across a group of rows in $S_{i,j}$. We pool across all the L rows in $S_{(i,j)}$ and get one value per filter (or feature map). We perform this operation for all the filters. Thus, overall, we obtain a feature rich representation of size z per sentence in $S_{(i,j)}$. We perform the convolution operation for all the sentences independently and obtain context representation by concatenating the sentence representations in the same sequence. Recall that S_i contained representations of $2K + 1$ sentences. Thus, overall the instance S_i is now represented by a sequence TS_i with $2K + 1$ units each of size z. TS_i is the output at Step 4 of Fig. 1.

We use shared filters for the *left-context* and *right-context* because: (1) It reduces the number of trainable parameters drastically making it easier for the model to train. (2) The representation vectors generated for both the *right-context* and *left-context* have the same semantics and lie in the similar vector space.

Stacked BiLSTMs with Attention. The problem could have been modeled as a sequence to sequence label generation task where each training sample is a whole document. But this model would be difficult to generalize for variable document length. Also, LSTMs have been shown to work well for shorter sequences. Hence, we first used CNNs to generate sentence embeddings and then use BiLSTM network on a smaller sequence that consists of only the main sentence and its neighbors.

To obtain a unified rich feature representation, we use Attention Bidirectional Long-Short Term Memory Network (Attention-BiLSTM) on top of this sequence TS_i. LSTMs [9] have been shown to model sequences better than vanilla Recurrent Neural Networks (RNNs) for various NLP tasks. LSTMs keep memories to capture long range dependencies. These memory cells allow error messages to flow at different strengths depending on the inputs. LSTMs have the ability to control the flow of information that flows to the memory cell state by using structures called *gates*. The reader is referred to [9] for details about LSTMs. To obtain a unified context representation which has rich feature set, we use a bidirectional LSTM (BiLSTM). The resultant embeddings are the concatenation of the two embeddings obtained through a forward pass LSTM and a reverse pass LSTM, capturing information from both the directions.

As shown in Fig. 1, we model the sequence of size K for both the left and the right context parts of S_i using separate LSTM networks each having K such memory cells, intuition being, two sentences at equal distance from the middle sentence might not have similar effect on the sentence being a segment boundary.

Traditional sequence encoder architecture which uses stacked BiLSTMs, forces the encoder to capture the information in a single fixed length representation. This also has a drawback as the hidden state at h_t is dependent on h_{t-1} across consecutive layers in the stack, thus the hidden state h_0 will have a significant effect on the future states. To overcome this, we use two vertically stacked BiLSTMs followed by soft attention [29]. Attention allows the model to give more importance to certain set of sentences in the context while ignoring the others, effectively learning the focus points to better predict if a sentence forms a segment boundary. We introduce an attention vector, z_s and use it to measure the relative importance of the sentences in the context as follows. Let $H_i^{K\times sz}$ denote the output of the last BiLSTM layer. Here sz is the size of the BiLSTM output corresponding to a sentence in the context. As shown in Fig. 1, $H_i = \{h_1, \ldots, h_K\}$.

$$e_i^{K\times 1} = H_i^{K\times sz} \times W^{sz\times 1} + b_i^{K\times 1} \tag{3}$$

$$a_i = \exp(\tanh(e_i^T z_s)), \quad \alpha_i = \frac{a_i}{\sum_p a_p}, \quad v_i = \textstyle\sum_{j=1}^{K} \alpha_j h_j \tag{4}$$

The resultant context embedding, v_i, which is jointly learned during the training process captures the essential information from the context. We compute this context embedding for both the *left-context* and *right-context*. The merged vector corresponds to the output at Step 7 of Fig. 1.

Dense Fully Connected Layer with Softmax. The resultant embeddings obtained from a shared encoder for the *left-context* and *right-context* and the embedding for *mid-sentence* obtained from separate but similar encoder are passed to a dense fully connected layer followed by an output.

$$P(y|s_i) = \text{softmax}(W_h \times v_i + b_i), \qquad y_{pred} = \arg\max\, P(y|s_i) \tag{5}$$

For classification we have a softmax layer over the output vectors. Finally, we take arg max over the predicted probability distribution to generate predictions.

4 Experiments

In this section, we discuss datasets, metrics and parameter settings for our experiments. Source code and datasets are available at https://github.com/pinkeshbadjatiya/neuralTextSegmentation.

4.1 Datasets

Text segmentation is quite a subjective task making evaluation of the text segmentation systems challenging. Hence, we use standard benchmark datasets for evaluation. Figure 2 shows summary of statistics about the datasets.

1. **Clinical** [12]**:** Consists of a set of 227 chapters from a medical textbook. Each chapter is marked into sections indicated by the author which forms the segmentation boundaries. It contains a total of 1136 sections.
2. **Fiction** [11]**:** Consists of a collection of 85 fiction books downloaded from Project Gutenberg. Segmentation boundaries are the chapter breaks in each of the books.
3. **Wikipedia:** Consists of randomly selected set of 300 documents having an average segment size of 26. These documents widely fall under the narrative category. Each document is divided into sections as marked in the original XML dump of the website. We use these section markers to create a segmentation boundary.

		Clinical	**Fiction**	**Wikipedia**
#Documents		227	85	300
#Sentences or #Samples		31868	27551	58071
Segment Length	Mean	35.72	24.15	25.97
	Std Dev	29.37	18.24	9.98

Fig. 2. Statistics of various datasets used for performance evaluation

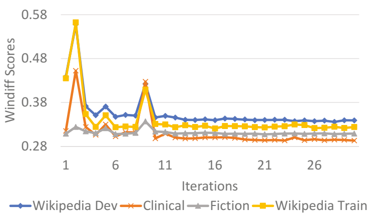


Fig. 3. Variation of windiff scores wrt iterations for various datasets

4.2 Metrics

We evaluate the performance of the model with respect to two metrics: Pk [1] and WinDiff [17]. Both the metrics use a sliding window of fixed size w over the document and compare the hypothesized segments with the reference ones. The window size, k, is generally set to half the gold-standard segmentation length [1]. Pk is the probability that two segments drawn from a document are incorrectly identified as belonging to the same segment. Windiff moves a sliding window across the text and counts the number of times the hypothesized and reference segment boundaries are different within the window. Counts are scaled to obtain probability values. Both Pk and Windiff thus lie between 0 and 1 and an algorithm that assigns all boundaries correctly receives a score of 0. WinDiff is considered a better measure than Pk as the Pk metric suffers from multiple issues as described in [17]. Pevzner et al. [17] proposed WinDiff as an update to the Pk metric. *Both the metrics are a loss measure. The lower the score, the better.*

4.3 Model Parameters and Training

For the purpose of training, we randomly select a set of pages from Wikipedia. We use section splits as our segment splits for training the model and skip the section headers, considering only the section content for training. We perform a 80-20 training-development split to obtain the training and development datasets. Figure 3 shows variation of WinDiff scores on the Wikipedia development dataset and other datasets as training progresses. The figure suggests that training converges well after 30 iterations, hence we fix number of iterations as 30. We trained our model on ∼270 documents from a sample of the Wikipedia corpus, creating ∼49k training sentences/samples. The average segment size of the whole Wikipedia corpus is around 9 sentences, while the test datasets have higher segment sizes (Fig. 2). We filter the documents that have average segment size less than 20 which results in training set having average segment length of 25. We train our model in batches of size 40. We set the context size, K, to 10 for all the experiments mentioned in Table 1.

The training dataset class distribution is heavily skewed with about 92% samples belonging to class 0. Hence, we use weighted-binary-cross-entropy as our loss function to penalize the classifier more heavily on mis-classification of a segment boundary. The loss function is defined as $loss = -\frac{1}{N}\sum_{i=1}^{N}(t\log(o) + \frac{f_1}{f_0}(1-t)\log(1-o)))$ where t and o are the target and the predicted outputs respectively. f_0 and f_1 are the frequencies of class 0 and class 1 respectively.

We use 'AdaDelta' [33] as the optimizer and use dropouts of 0.2–0.3 for input and recurrent gates in the recurrent layers. We also use dropouts of 0.3 after the dense fully connected layers to prevent over fitting on the training dataset. We use filters of sizes $\{2, 3, 4, 5\}$ with 200 filters for each of the sizes. The recurrent layers have 600 neurons.

4.4 Comparison with Other Baseline Methods

We compare the performance of our proposed model against various competitive baselines, four basic neural models, and three BiLSTM model variants. Each of those models help us understand contributions of the various components of the proposed model. Table 1 shows the summary of the results obtained using various models on all three benchmark datasets. In the following, we describe the baseline systems in brief.

1. **U&I** [25]**:** It is a probabilistic framework based on maximizing the compactness of the language models induced for each segment using ideas similar to the noisy channel and minimum description length methods.
2. **MinCut** [12]**:** This method treats text segmentation as a graph-partitioning task aiming to optimize the normalized-cut criterion. It simultaneously optimizes the total similarity within each segment and dissimilarity across segments.
3. **BayesSeg** [4]**:** This method models the words in each topic segment as draws from a multinomial language model associated with the segment. Segmentation is obtained by maximizing the observation likelihood in such a model.
4. **APS** [11]**:** Affinity Propagation for Segmentation receives a set of pairwise similarities between data points and produces segment boundaries and segment centers. Data points which best describe all other data points within the segment are considered segment centers. APS iteratively passes messages in a cyclic factor graph, until convergence.
5. **PLDA** [18]**:** PLDA is a generative model that uses Bayesian inference to simultaneously address the problems of topic segmentation and topic identification. It chains a set of LDAs by assuming a Markov structure on topic distributions.
6. **TSM** [2]**:** Structured Topic Model is a hierarchical Bayesian model for unsupervised topic segmentation. It uses an MCMC inference to split/merge segment(s).

For all these baseline algorithms, we use the publicly available source codes. We also fine tune the parameters, using the scripts provided by the authors, for our experiment on the Wikipedia dataset to get the optimal set of parameters. We could not perform some experiments where source codes were not available publicly. We mark those instances with NA. We also compare the performance on "Random" baseline where we place the segment boundaries randomly in the text. Part A of Table 1 shows the observed results from our experiments on these baselines.

To understand the contribution of various components in our proposed model we compare the performance of other neural models as well without using any context information. We discuss these neural models in brief below. Part B of Table 1 presents the performance for four such neural architectures.

1. **Perceptron:** We encode the sentence using mean of the word2vec representations of the corresponding words and then learn a 5-layered perceptron.

2. **LSTM:** We represent each sentence using a sequence of words and then learn a combined dense representation for each sentence in the vector space using the word2vec embeddings for words.
3. **Stacked-LSTM:** This model is similar to LSTM, except that it provides more flexibility at the cost of more trainable parameters.
4. **CNN:** We use the CNN based sentence representations obtained by convolving multiple variable length filters with the word embeddings to obtain rich feature representations for each sentence.

We also compare the performance of our proposed model with other BiLSTM based neural models. Each of these models obtain specific sentence representations which are then passed to a BiLSTM architecture to obtain context representations. Part C of Table 1 presents the performance for these neural architectures besides the proposed method, CNN+Attn-BiLSTM.

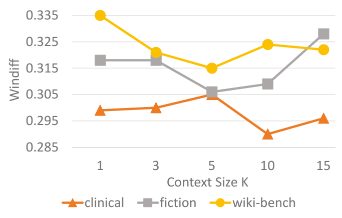
1. **MeanBoW-BiLSTM:** We use mean of all the word vectors to obtain a sentence representation, which along with its neighboring context, is passed to a stacked BiLSTM encoder architecture to obtain the context representations.
2. **TFIDF MeanBoW-BiLSTM:** To obtain a sentence embedding, we compute weighted mean of the word2vec word embeddings where TF-IDF (Term Frequency-Inverse Document Frequency) scores are used as weights. TF-IDF scores capture the relative relevance of a particular word in the sentence.
3. **CNN-BiLSTM:** We use the CNN based sentence representations obtained by convolving multiple variable length filters with the word embeddings to obtain rich feature representations for each sentence. There is no attention layer in this method.

Table 1. Accuracy comparison of the proposed approach with competitive baselines. **Lower values are better.** Experiments marked with **NA** could not be performed due to non-availability of publicly available source codes. Some of the cell values have been directly taken from respective papers, if they mentioned them for the same (method, dataset) pair.

	Model	Clinical		Fiction		Wikipedia	
		WinDiff	Pk	WinDiff	Pk	WinDiff	Pk
Part A: Competitive baselines	U&I [25]	0.376	0.370	0.459	0.459	0.368	0.368
	MinCut [12]	0.382	0.368	0.405	0.371	0.389	0.364
	BayesSeg [4]	0.353	0.339	0.337	**0.278**	0.390	0.359
	APS [11]	0.399	0.396	0.480	0.451	0.380	0.392
	PLDA [18]	0.373	0.324	0.430	0.361	NA	NA
	TSM [2]	0.345	**0.306**	0.408	0.325	NA	NA
	Random	0.459	0.441	0.510	0.475	0.486	0.480
Part B: Neural models without context	Perceptron	0.338	0.357	0.336	0.335	0.421	0.415
	Stacked-LSTMs	0.381	0.393	0.329	0.394	0.437	0.420
	LSTMs	0.486	0.471	0.366	0.417	0.508	0.455
	CNN	0.309	0.329	0.314	0.386	0.363	0.380
Part C: Context based neural models	MeanBoW + BiLSTM	0.349	0.365	0.319	0.389	0.405	0.398
	TF-IDF MeanBoW + BiLSTM	0.345	0.366	0.328	0.382	0.382	0.392
	CNN + BiLSTM	0.334	0.316	0.331	0.324	0.378	**0.328**
	CNN + Attn-BiLSTM	**0.294**	0.318	**0.308**	0.378	**0.315**	0.344

5 Analysis of Results

In this section we analyze the results of our experiments and compare their performance with the existing non-neural as well as neural baselines. We also briefly discuss about the choice of metrics in Subsect. 5.4.



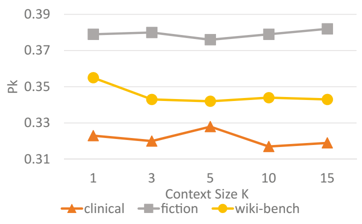


Fig. 4. Variation of WinDiff scores on various datasets with varying context size K

Fig. 5. Variation of Pk scores on various datasets with varying context size K

5.1 Comparison with Baseline Models

Table 1 compares the performance of the proposed Attention-based model on various benchmark datasets with other existing models. We observe that the use of Attention-based supervised models provide a performance improvement over other methods across all the datasets on the Windiff metric, and compares well with the best method on the Pk metric.

The proposed model has additional benefits with respect to runtime performance. Once the training is finished, during the prediction phase, our model takes on an average 0.09 s on a batch of 40 sentences on GeForce GTX 1060 GPU which is much faster than other methods in Part A of Table 1 which take time to the order of minutes to days during prediction phase as most of the computation takes place during that time.

5.2 Comparison with Neural Models

It is important to note that without context the neural models (Part B) sometimes perform worse than the baseline models (Part A). All of the variants of LSTMs in Part C are better than the context-unaware LSTM model in Part B. In Part C, we note that the use of TFIDF-weighted-Mean BoW word2vec embeddings for representing sentences (TFIDF MeanBoW + BiLSTM) only provides a slight improvement. We conclude that for the task of text segmentation, TF-IDF features do not add additional information as compared to the word-embeddings.

Our experiments with the CNN model provides us with good results compared to other model variants overall. Results in Part B show improved performance even without using any context information with the use of CNN for obtaining sentence representations. Our experiments with CNNs along with BiLSTM do not show much improvement in the results on the Windiff metric, though they show significant improvement on the Pk metric results across all the datasets encouraging us to use BiLSTM as part of our proposed model. The use of Attention further improves the results by a significant margin across all the datasets.

We also observe improved performance with the use of Attention on the neighboring context. Use of soft Attention [29] allows the model to focus on certain regions more than others in order to generate better context representations. Using the attention layer shows an increase in Windiff performance by 4%, 2.3% and 6.3% on Clinical, Fiction and Wikipedia datasets respectively over the CNN+BiLSTM model. Overall, the proposed model shows an improvement of 5.1%, 10% and 6.5% on Windiff metric on Clinical, Fiction and Wikipedia datasets respectively, and improvement of 3.1% on Pk metric on the Wikipedia dataset compared to the existing competitive baselines.

5.3 Varying Context Size K

We also experiment with the variation in context size K on all the datasets and report the results in Figs. 4 and 5. We train our model on the same training dataset for 30 epochs and report the results with variable context sizes. We observe a decreasing trend (recall low Windiff scores are good) in the WinDiff scores as Context size increases which gradually starts increasing as context size grows. All the three datasets follow this trend for the Windiff metric, while only the Fiction and Wikipedia datasets have shown similar results for Pk metric. For Clinical dataset, we lose a lot of domain specific information while converting words to vectors using the word-embeddings as word2vec is not very rich in domain specific information. This leads to poor results on the Windiff metric with low context-size. As context size grows, it gathers enough information to correctly classify the segment boundaries.

5.4 Implication of Windiff and Pk Metric Results

Since the models are trained on the Wikipedia Dataset, it is focused towards presenting an evenly spread out segmentation of the paragraph, as learned from the Wikipedia documents, which often have a uniform section distribution. The Clinical and Fiction datasets both have quite a significant number of segments with less number of sentences (as evident from the very high standard deviation reported in Fig. 2) resulting in less number of predicted segment boundaries. Assuming a window of size λ, each miss of segmentation boundary will produce a false negative. Each such false negative will receive a total of λ penalties. Since the model often results in less number of segmentations, it often receives higher number of penalties than expected, resulting in higher Pk scores. Hence, as also seen in Table 1, baseline neural models consistently perform poorly on the Pk metric. But the Pk scores for neural models on Wikipedia dataset are comparable to the state-of-the-art methods.

Choice of WinDiff over Pk: Pk metric suffers from multiple issues due to which it does not provide a good measure of the hypothesized segmentations. These issues are covered in detail by Pevzner et al. [17], motivating us to use the WinDiff metric as our primary measure for comparing performance with the existing models and baselines. We still report the results in both the metrics as they might provide beneficial information to the readers.

6 Conclusions

In this paper, we studied the problem of text segmentation from a neural perspective. We presented a model which first learns rich features for every sentence using Convolutional Neural Networks followed by sequential learning using temporal data. Finally, we also learn focus on various sentences in the context using the attention layer. We performed extensive experiments to compare against well-established non-neural baselines, as well as against recent neural models. Experimenting with three different datasets, we empirically proved that our proposed model provides lowest Windiff loss with very little supervision, and with low execution times.

In the future, we plan to test this model on non-English datasets, especially morphologically rich languages where huge datasets are not available for training the model and most of the contextual information is captured at the word level rather than at the sentence level.

References

1. Beeferman, D., Berger, A., Lafferty, J.: Statistical models for text segmentation. Mach. Learn. **34**(1), 177–210 (1999)
2. Du, L., Buntine, W.L., Johnson, M.: Topic Segmentation with a structured topic model. In: HLT-NAACL, pp. 190–200 (2013)
3. Du, L., Pate, J.K., Johnson, M.: Topic segmentation in an ordering-based topic model. In: AAAI, pp. 2232–2238 (2015)
4. Eisenstein, J., Barzilay, R.: Bayesian unsupervised topic segmentation. In: EMNLP, pp. 334–343. ACL (2008)
5. Galley, M., McKeown, K., Fosler-Lussier, E., Jing, H.: Discourse segmentation of multi-party conversation. In: ACL, pp. 562–569 (2003)
6. Grosz, B., Hirschberg, J.: Some intonational characteristics of discourse structure. In: Proceedings of the Second International Conference on Spoken Language Processing (1992)
7. Hajime, M., Takeo, H., Manabu, O.: Text segmentation with multiple surface linguistic cues. In: ACL, pp. 881–885 (1998)
8. Hearst, M.A.: TextTiling: segmenting text into multi-paragraph subtopic passages. Comput. Linguist. **23**(1), 33–64 (1997)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
10. Joty, S., Carenini, G., Murray, G., Ng, R.T.: Supervised topic segmentation of email conversations. In: ICWSM (2011)
11. Kazantseva, A., Szpakowicz, S.: Linear text segmentation using affinity propagation. In: EMNLP, pp. 284–293. ACL (2011)

12. Malioutov, I., Barzilay, R.: Minimum cut model for spoken lecture segmentation. In: ACL, pp. 25–32 (2006)
13. Misra, H., Yvon, F., Jose, J.M., Cappe, O.: Text segmentation via topic modeling: an analytical study. In: CIKM, pp. 1553–1556. ACM (2009)
14. Mitrat, M., Singhal, A., Buckley, C.: Automatic text summarization by paragraph extraction. In: Intelligent Scalable Text Summarization (1997)
15. Nguyen, V.A., Boyd-Graber, J., Resnik, P.: SITS: a hierarchical nonparametric model using speaker identity for topic segmentation in multiparty conversations. In: COLING, pp. 78–87 (2012)
16. Oh, H.J., Myaeng, S.H., Jang, M.G.: Semantic passage segmentation based on sentence topics for question answering. Inf. Sci. **177**(18), 3696–3717 (2007)
17. Pevzner, L., Hearst, M.A.: A critique and improvement of an evaluation metric for text segmentation. Comput. Linguist. **28**(1), 19–36 (2002)
18. Purver, M., Griffiths, T.L., Körding, K.P., Tenenbaum, J.B.: Unsupervised topic modelling for multi-party spoken discourse. In: ACL, pp. 17–24 (2006)
19. Reynar, J.C.: Statistical models for topic segmentation. In: ACL, pp. 357–364 (1999)
20. Riedl, M., Biemann, C.: TopicTiling: a text segmentation algorithm based on LDA. In: ACL Student Research Workshop, pp. 37–42 (2012)
21. Rush, A.M., Chopra, S., Weston, J.: A neural attention model for abstractive sentence summarization. In: EMNLP, pp. 379–389. ACL (2011)
22. Sakahara, M., Okada, S., Nitta, K.: Domain-independent unsupervised text segmentation for data management. In: ICDMW, pp. 481–487 (2014)
23. Sheikh, I., Fohr, D., Illina, I.: Topic segmentation in ASR transcripts using bidirectional RNNs for change detection. In: IEEE Automatic Speech Recognition and Understanding Workshop (2017)
24. Tür, G., Hakkani-Tür, D., Stolcke, A., Shriberg, E.: Integrating prosodic and lexical cues for automatic topic segmentation. Comput. Linguist. **27**(1), 31–57 (2001)
25. Utiyama, M., Isahara, H.: A statistical model for domain-independent text segmentation. In: ACL, pp. 499–506 (2001)
26. Dijk, T.A.V.: Episodes as units of discourse analysis. In: Analyzing Discourse: Text and Talk, pp. 177–195 (1982)
27. Wang, L., Li, S., Xiao, X., Lyu, Y.: Topic segmentation of web documents with automatic cue phrase identification and BLSTM-CNN. In: Lin, C.-Y., Xue, N., Zhao, D., Huang, X., Feng, Y. (eds.) ICCPOL/NLPCC -2016. LNCS (LNAI), vol. 10102, pp. 177–188. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50496-4_15
28. Wang, L., Li, S., Lyu, Y., Wang, H.: Learning to rank semantic coherence for topic segmentation. In: EMNLP, pp. 1340–1344. ACL (2017)
29. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y.: Show, attend and tell: neural image caption generation with visual attention. In: ICML, pp. 2048–2057 (2015)
30. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: HLT-NAACL, pp. 1480–1489 (2016)
31. Yin, W., Schütze, H., Xiang, B., Zhou, B.: ABCNN: attention-based convolutional neural network for modeling sentence pairs. In: ACL, pp. 259–272 (2016)
32. Yu, J., Xiao, X., Xie, L., Chng, E.S.: Topic embedding of sentences for story segmentation. In: APSIPA ASC (2017)
33. Zeiler, M.D.: ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)