Domain-independent Text Segmentation

Mona Fadaviardakani
UBC CS Department
mfadavi@cs.ubc.ca

Abstract

Identifying segment boundaries in textual documents is one of the fundamental problems in Natural Language Processing (NLP) field. Traditional solutions for the text segmentation are suffering from manual feature engineering, long run-time, and huge memory requirement. Most proposed neural models suffer from challenges such as sparsity of segment boundaries and variability of the output size vocabulary. In this project, I considered these main problems in the architecture of my model and focused on proposing the neural architecture model which not only covered broad range of samples, but also considered other segment boundary positions in its decision. To achieve that, I chose one of the previous works as my baseline model and try to improve it by using the multi-layer pointer network. I used datasets containing different domains and different statistics to train and test my model. The result shows, my model worked in case we have longer and more sparse segments.

1 Introduction

Text segmentation, the problem of separating the documents into coherent segments based on their semantic similarity, has received a lot of attention due to its usage in many NLP tasks. Detailed segmentation of the document helps better representation of its structure and thus can be useful in diverse tasks such as automatic summarization, question-answering, and discourse analysis.

We can consider this task from two different points of view. At the sentence-level, it often refers to the problem of identifying the sentence positions at which topic changes in the stream of the text, also known as topic segmentation. At a more detailed level, it can also be considered as the fine grained segmentation of a sentence into a sequence of elementary discourse units called EDU segmentation (Marcu, 2000).

According to (JJing Li and Joty, 2018), most proposed neural models suffer from challenges such as sparsity of their segment boundaries and variability of their output sequence.

There are different approaches which can be considered in order to treat this task. Both topic and EDU segmentation can be treated as the sequence-labeling/binary classification problem which predicts yes/no boundary tags at either sentence or word level. The ”yes” tag means that the sample starts a new segment and ”no” tag means that sample is still belonging to the previous segment. One way to resolve the sequence classification task is using popular seq2seq encoder-decoder models. However, the drawback of these approaches is that the output dictionary is fixed and is not dependent on the input sequence. In addition, binary classification approaches are only considered one segment classification at a time and cover limited range of samples for binary decision.

In this project, I focused on building a neural domain-independent approach for partitioning task into coherent regions at the sentence level. I treated this task as the prediction of the segmentation positions in the input sequence and purposed the architecture which resolves both variability of output sequence and dependency on the input sequence to resolve the sparsity problem. Also, I

made use of attention in order to be able my model to automatically tune the degree of dependence on the context, as this dependency may be different in diverse domains.

2 Related Work

Text Segmentation-There are both supervised and unsupervised methods which are proposed to tackle text segmentation problem. Unsupervised methods (Riedl and Biemann, 2012) mostly need huge memory, long run-time, and can not generalize well across different text structures (cause-effect, problem-solution ,and etc) and writing styles (informative,descriptive ,and etc.). In addition, supervised methods (Pei-Yun Hsueh and Renals, 2006) (ugo Hernault and Ishizuka, 2010) (Shafiq Joty and Codra, 2015) often require domain expertise and feature engineering which is costly in terms of data annotations.

There are little neural works have been done so far for this problem. (Omri Koshorek and Berant, 2018) proposed the hierarchical architecture of two sub networks of sentence embedding and segment prediction. The first sub network (at the word-level) is composed of two bi-directional LSTM layers which takes the words of a sentence as the input and passed them through the max-pooling layer which is responsible to generate the sentence embedding. The next sub network takes the sequence of sentence-level embeddings and fed them through the two bidirectional LSTM layers. These outputs are then passed through the fully connected layer and the softmax layer in order to generate the cut-off probability for each sentence.

(Badjatiya and Varma, 2018) proposed an attention-based bidirectional LSTM model which benefits of the combination of CNN and bi-LSTM for learning the sentence embeddings and attention for assigning different importance of the context to every sentence for segmentation. This paper is based on the key intuition that every sentence has the right and left context including K sentences before and K sentences after it.

(JJing Li and Joty, 2018) used bi-directional RNN to encode the input sequence and another uni directional RNN as the decoder together with a pointer network to determine the text boundaries. This paper proposed general model for both sentence and word level segmentation, therefore text boundaries can be either word or sentence depending on the input sequence.

Pointer Network- Pointer network is a new neural architecture which learns the conditional probability of an output sequence with elements that are discrete tokens corresponding to positions in an input sequence (Oriol Vinyals and Jaitly, 2015). This architecture enables to resolve variability in the output size since the number of target classes in each step of the output depends on the length of the input, which is variable. In order to have variable size output dictionaries, pointer networks are using a mechanism of neural attention that works as a pointer to select a member of the input sequence as the output. In this project, I benefit from pointer network architecture to have variable size for segment positions outputs.

3 System Overview

In this section, I proposed a new neural network architecture which is affected by the previous works (JJing Li and Joty, 2018),and (Badjatiya and Varma, 2018) as my base models. In the followings, first the dataset and preparation steps are explained. Further, I discussed the base models and finally, my approach is presented in the contribution section.

3.1 Dataset

I chose the standard benchmark datasets (Badjatiya and Varma, 2018) for the training and evaluation of my model as it is coming in different domains and have different statistics for each dataset. My datasets are consisting of clinical, fiction, biography and Wikipedia random texts. The clinical datasets are containing of 227 chapters with 1136 sections, and average segment length of 35.72. The fiction dataset are containing of 85 fiction books and 27551 sentences with the average segment length of 24.15. The segments are considered as the chapter breaks in each of these books instead of sections in each chapter. The Wikipedia texts are containing of 300 random documents, 2265 paragraphs and 58578 sentences having an average segment of 25.97. These texts are mostly has narrative writing style and their sections are defined with HTML markers. The biography dataset is also contains 11 chapters, 298 paragraphs and 2285 sentences with the average segment length of 7.66.

The total numbers of training samples is about 119775 samples. The advantage of this set of

dataset is the diversity of segment lengths and types defined in different domains which makes the evaluation more realistic. A detailed statistics of datasets are shown in Table 1.(Badjatiya and Varma, 2018)

3.2 Preprocessing

Prior to the embedding extraction, I tokenized texts, removed punctuation marks and non alphabetical characters, converted all words to lowercase, and eliminated the stop words. I also used paddings for both the sentences and the batches using in training of my models in order to have fixed size for all parts.

3.3 Model

In this section, first I briefly discuss both of my baseline models and then go through my proposed architecture.

3.3.1 First Baseline Model

(Badjatiya and Varma, 2018) addressed the text segmentation as a binary classification problem which predicts whether the sentence denotes the beginning of a new text segment or not.

Figure 1 illustrates the base model architecture and 8 steps which are involved in this process . The input of this model is the mid sentence, and the left and right contexts. The same operations are done on each of these contexts (through step 1-6) and finally all of these outputs are merged in order to get the classification result. By using the same operations, the number of trainable parameters are reduced and we have the similar semantics for the different contexts. At the first step, the embedding matrix for the mid sentence and for batch of left and right context sentencs are obtained by using the 300D word2vec embedding. The word2vec is trained on Google News dataset. And for missing words in vocabulary, word-based lemmatization is applied.

At the next step, 1D convolution operations with Z number of filters are performed on each sentence embeddings in the batch sentence representation. the output of this step is the Z feature vectors for each of k sentences existing in the batch contexts. At step 3, maxpooling is applied to get the max feature value over all z feature vectors of each sentence. This is performed on all of the batch sentences independently resulted in having k convoluted features of size Z . These features are concatenated in step 4 and fed through the stacked

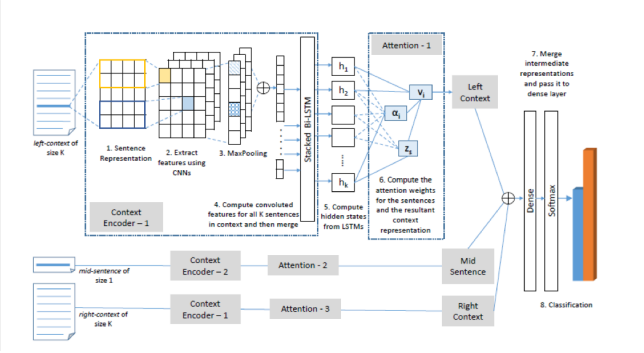

Figure 1: The First Baseline Model Architecture

bi-LSTM with K memory cells. The authors decided to use LSTM to capture the sequence dependencies and also make it bi-directional to obtain the concatenated embeddings resulted from both forward and reserves pass LSTMs . They claimed this model give them the best performance comparing to their other trials. Next, the attention vector z_s is used to give more importance to set of sentences . It helps the model to learn the focus points and raises the model performance. If we consider sz as the size of BiLSTM output for each sentence, and H_i^{K*sz} as the output of last BiLSTM layer for the K sentences, we compute the context embedding v_i as follows:

$$e_i^K = H_i^{K*sz} * W^{sz} * b_i^K \tag{1}$$

$$a_i = exp(tanh(e_i^T * z_s)) \tag{2}$$

$$\alpha_i = a_i / \sum_p a_p \tag{3}$$

$$v_i = \sum_{j=1}^{k} \alpha_j h_j \tag{4}$$

Finally, at the step 6 outputs coming from mid, right, and left contexts are getting merged and passed through a dens fully connected layer following the softmax layer which transforms the output from the hidden size to the binary yes, no tags.

There are two main drawbacks associated with this approach:

- It is quite helpful to capture the dependencies of input sentence when the boundaries are sparse. (Badjatiya and Varma, 2018) only focus on one segment boundary at a time and does not consider other segment boundary positions.

	# documents	#samples	#segments	average of segment length
Clinical	227	31868	1136	35.72
Wikipedia	300	58071	2265	25.97
Fictions	85	27551	1245	24.15
Biography	11	2285	298	7.66

Table 1: Overview of Datasets Statistics

- In order to do binary classification for each sentence, the base model only considers 2K+1 sentences and does not consider the broader range of sentences.

3.3.2 Second Baseline Model

(JJing Li and Joty, 2018) addressed text segmentation problem in both sentence and world levels and defined the problem as the inputs are the sequence of sentences and outputs are positions of the segment boundaries in the input sequence. Figure 2 illustrates the model architecture consisting of three main components of encoding, decoding, and pointing.

In the encoding phase, bi-directional GRUs is used to capture sequential dependencies of the input sequence. Because the number of boundaries in the output vary with the input, the RNN-based models are also used to decode the output.

Since this approach aims to capture other segment boundary positions and also resolve the problem of variability in the output size, it used multi-layer pointer network.

At each step, the decoder takes a start unit of a segment in the input sequence and computes a distribution over the possible positions in the input sequence for a possible segment boundary. If the input sequence contains M boundaries, the decoder produces hidden states in d $\in \mathrm{R}^{M*H}$ which H denotes the dimension of the hidden layer.

At each step, the pick of an output distribution is chosen as the segment boundary and the next starting point is passed through the decoder. In order to compute distribution over possible positions in the input sequence, pointer networks are using a mechanism of neural attention that works as a pointer to select a member of the input sequence as the output. This attention formula for each step m is the following:

$$u_j^m = v_t^t an(W_1 h_j + W_2 d_m) for j \in (m, ...M \tag{5}$$

$$p_y(m|x_m) = softmax(u_m) \tag{6}$$

Parameter j is the possible position in the input sequence and the softmax over u_j^m is indicating the probability of U_j as a boundary, given the start unit U_m.

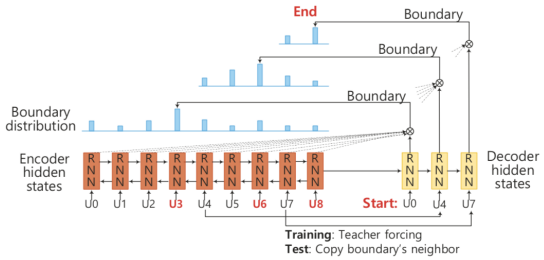

Figure 2: The 2nd Baseline Model Architecture

3.3.3 Contribution

Figure 3 shows the architecture of my proposed model. The input of my model is the desired sequence of M size. For every sentence in this sequence, the $2k + 1$ embeddings are computed and passed through bidirectional GRUs. These embeddings are resulted from the step 6 of the first baseline model which is pre-trained by wikipedia dataset and its two last layers (softmax and dense) are removed.

These embeddings are the rich representation for every sentence and the advantage of using these embeddings is broader coverage of other sentences.

The batch with M size of these representations are passed through the multi-layet pointer network. Therefore, my approach has two main hyperparameters of context size k and the sequence size of M.

At each step, the decoder takes the ground-truth starting sentence of the segment (teacher forcing) as the input and compute the output distributions over the rest of sentences. For example, like Fig 3, consider the sequence containing from U0 to U8 and U3, U6 as boundaries. First,it starts from the fist starting segment sentence, S0, and compute the output distributions over S1 to S8. At the next step, it starts with the next sentence starting

	reference PK	obtained PK	reference windiff	obtainded windiff
Clinical	0.318	0.630	0.794	0.938
Fictions	0.378	0.4789	0.308	0.4210
Biography	0.38501	0.38518	0.258545	0.258432

Table 2: Comparison results between the reference model and my approach

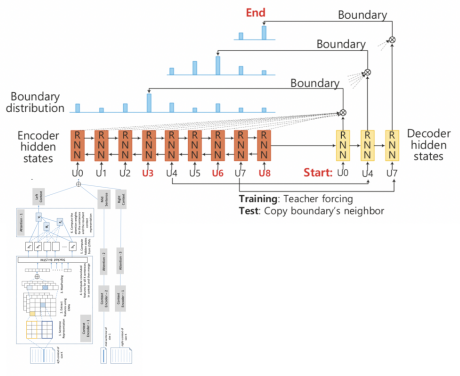

Figure 3: My Model Architecture

the second segment, S4, and compute the output distribution over the rest (S4 to S8). And finally, the last start point, S7, are passed and distribution are computed untill we reach the end of sequence. Multi-layer pointer network allowed us to consider other segment boundary positions and has the variable output size which are quite helpful in addressing text segmentation problem. In the next section, Implementation details and evaluation results are explained.

4 Experiments

4.1 Implementation Details

Two neural network models are implemented by using Keras framework. The learning rate which is used for training the first network is 0.1 and I used the context size of 10 and the batch size of 64. The loss function which is used here is binary cross entropy. The output of concatenation layer has 2700 dimensions resulted in concatenating mid,left and right contexts. The learning rate which is used for training the second network is 0.001 and I used the max-input-sequence-length of 10 and the batch size of 128. In order to get the input of my second model, I set the batch size of first neural model to 10 and get (10,2700) dimensions as the input sequence of my second neural model. The negative log likelihood is used as the loss for my second network. In order to do teacher forcing, at each step of pointer network, I trained the second model by supplying the ground-truth start unit to decoder. I trained both models with 5 epochs and used wikipedia dataset for training both models and other three datsets for the testing part.

4.2 Evaluation

There are two most frequently used metrics for text segmentation: Pk and windiff. Both metrics use window of size K.
Pk only considers whether end points of the window contain a boundary or not, and do not consider the number of boundaries between the two end points. However, windiff covers this and penalize both false positives and false negatives equally (by moving sliding window across the text) (Pevzner and Hearst.). I used the results of (Badjatiya and Varma, 2018) for the comparison with my results which is illustrated in Table 2.
The result shows that my approach can effectively improve performance on clinical dataset, and not that much effective in biology dataset in comparison with the results obtained from the first baseline model (Badjatiya and Varma, 2018). Since, clinical dataset contains longer (average segment length is 32.75) and more sparse segments (1136 segments out of 31868 samples), my approach can effectively help on improving the performance. While in case of biograpgy data, segments are smaller (average segment length is 7) and it is not that much sparse, therefore the performance is not effectively changed.

5 conclusion

In this project, I tried to find optimal solution on how to predict the segment boundary position in the input sequence of samples. My proposed architecture addressed both issues of variability of the outputs sequence and sparsity in segment positions. I used the sequence of the rich context representation of samples as the input to multi-layer pointer network. The evaluation shows my approach can raise the performance specially when the average segment length of the dataset are large

and dataset include more sparse segments.

6 Discussions

In order to do this project, I followed the schedule coming below:

- Do literature review.
- Conduct data collection and analysis.
- Implement data preprocessing, data loader, and the sample handler.
- Create project template with Keras.
- Implement my first baseline model,train it, and remove the last two layers.
- Implement next baseline model, the multi-layer pointer network.
- train the model with pre-trained embeddings obtained from the first baseline model (for each sample).
- test my model on other datasets.
- Write my research final report.

In this project, I learned about pointer networks, how to implement a designed model from scratch to end, and use the output of my own pre-trained model to another.

References

L. J.; Gupta M.; Badjatiya, P.; Kurisinkel and V Varma. 2018. Attention-based neural text segmentation. *In ECIR18*.

Danushka Bollegala ugo Hernault and Mitsuru Ishizuka. 2010. A sequential model for discourse segmentation. *In CICLing*.

Aixin Sun JJing Li and Shafiq Joty. 2018. Segbot: A generic neural text segmentation model with pointer network. *In Proc. IJCA*, pages 4166–4172.

Daniel Marcu. 2000. The theory and practice of discourse parsing and summarization. In *MIT press*.

Noam Mor Michael Rotman Omri Koshorek, Adir Cohen and Jonathan Berant. 2018. Text segmentation as a supervised learning task. *In arXiv preprint arXiv:1803.09337*.

Meire Fortunato Oriol Vinyals and Navdeep Jaitly. 2015. Pointer network. *In NIPS*.

Johanna D Moore Pei-Yun Hsueh and Steve Renals. 2006. Automatic segmentation of multiparty dialogue. *In EACL*.

Pevzner and M. Hearst. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*.

Martin Riedl and Chris Biemann. 2012. Topictiling: a text segmentation algorithm based on lda. *In In ACL*.

Giuseppe Carenini Shafiq Joty and Raymond T Ng. Codra. 2015. Automatic segmentation of multiparty dialogue. *Computational Linguistics*.