Literature Survey: Text Segmentation

Debayan Bandyopadhyay
Pushpak Bhattacharyya
CFILT, Indian Institute of Technology Bombay, India
{debayan, pb}@cse.iitb.ac.in

Abstract

Text Segmentation is an application of Natural Language Processing used for splitting a piece of text into small meaningful paragraphs. Since it is hard to define criteria on which the segmentation can be done, researchers have come up with many approaches. Discourse phenomena, *Cohesion* and *Coherence* are majorly used to detect potential positions in text. But there are works that utilize patterns for text segmentation. This report is a brief introduction to the major works in the field of text segmentation.

1 Introduction

When people learn how to write, they are advised to keep sentences conveying the same idea in a single paragraph. This makes the paragraph coherent, and the text easy to read. Complementarily, the advice is given to start a new paragraph when there is a major change in idea. Inexperienced writers face difficulty in creating paragraphs. Too many paragraphs break the flow of reading, while too few create confusion. Paragraphing thus is an art, an exercise in striking a balance between stop-and-start-of-ideas and crowding-of-ideas.

Paragraphing *aka, Text Segmentation* (which term we will henceforth use) is an application of Natural Language Processing for the creation of coherent and cohesive text. The input is a piece of text and the output, segments. The challenge is to ensure that each paragraph is meaningful and self-sufficient. We call a paragraph self-sufficient if it conveys a single idea. The contents of its neighboring paragraphs do not convey the same idea. Text Segmentation finds use in improving the readability of text and various Information Extraction tasks like Paragraph Retrieval (Dias et al., 2007) and Text Summarization (Chuang and Yang, 2000; Pourvali and Abadeh, 2012).

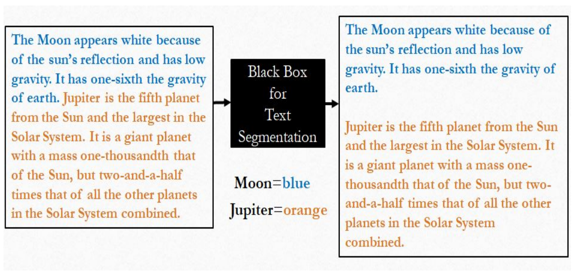

The Moon appears white because of the sun's reflection and has low gravity. It has one-sixth the gravity of earth. Jupiter is the fifth planet from the Sun and the largest in the Solar System. It is a giant planet with a mass one-thousandth that of the Sun, but two-and-a-half times that of all the other planets in the Solar System combined.
Black Box for Text Segmentation
Moon=blue
Jupiter=orange
The Moon appears white because of the sun's reflection and has low gravity. It has one-sixth the gravity of earth.
Jupiter is the fifth planet from the Sun and the largest in the Solar System. It is a giant planet with a mass one-thousandth that of the Sun, but two-and-a-half times that of all the other planets in the Solar System combined.

Figure 1: Black Box of Text Segmentation

2 Problem Statement

- Input: A piece of text
- Output: Text with paragraph splits

Figure 1 shows a black box for text segmentation, splitting the text based on Moon and Jupiter.

3 Metric for evaluation

Precision and Recall are not a good option for evaluating the performance of a text segmentation system. Using such metrics will impose an equal penalty on
i) a system producing paragraph boundary one position off from the actual position of the boundary, and
ii) a system producing paragraph boundary far off from the actual position of the boundary.

Since the metrics will be unfair to the systems, researchers have come up with better metric systems to evaluate the performance of text segmentation systems.

3.0.1 P_k Metric

Beeferman et al. (1999) came up with a new metric P_k. Before describing the intuition for the annotation, the following notations have to be explained: $\delta_x(i, j)$ is an indicator variable which returns 1 if sentence i and j are in different paragraphs. Otherwise, it returns 0. x denotes the case when the

indicator variable is used. The case can be for reference and hypothesis.
The intuition is that if two random sentences are taken from the text, and we check using the reference text, whether they belong to the same paragraph or not. Then we use the text with a hypothesized boundary generated by the text tiling algorithm to check whether the sentences belong to the same paragraph or not. If the output of the two is the same, no penalty is added. Otherwise, a penalty is added. This can be mathematically represented as follows:

$$Penalty_(i,j) = (\delta_{hyp}(i,j) \overline{\oplus} \delta_{ref}(i,j)) \quad (1)$$

P_k finds the penalty for all sentence pairs(i,i+k)

In simpler terms, P_k calculates the probability that two sentences k-sentence apart are misclassified to belong to the same paragraph by a model.

3.0.2 WindowDiff Metric

Though P_k metric turned out to be better alternative of precision, recall and f-score, Pevzner and Hearst (2002) found flaws of the P_k metric. Some of the flaws of the metric include:

- False-negative penalized more than false positive
- No penalty for the number of boundaries
- Error varies with segment size
- Near miss error penalized too much

To overcome the problems of the P_k metric, the authors propose their approach: WindowDiff.

The authors take a window of size w and count the number of boundaries present in the window segment produced by the text segmentation model, and the number of boundaries present in the window segment of the actual text. If the count of boundaries in both the segment is equal, then no penalization is imposed. Otherwise, the model incurs a penalization.

The mathematical representation for the penalizing is represented as:
$Windowdiff(ref, hyp) = \frac{1}{N-w}\sum_{i=1}^{N-w}(|b(ref_i, ref_{i+k})b(hyp_i, hyp_{i+k})| > 0)$
where b(i,j) denotes the number of boundaries between sentence with index i and sentence with index j.
$(|b(ref_i, ref_{i+k})b(hyp_i, hyp_{i+k})| > 0)$ is an indicator variable which gives 1 as output if the condition is true and otherwise it returns false.

4 Text Segmentation Datasets

4.1 WIKI727K and WIKI50

Koshorek et al. (2018) utilized the structure of Wikipedia articles and extracted the text to create datasets for text segmentation. We will refer to these datasets as corpus since they are collections of text having well-defined paragraph boundaries.

The authors created the WIKI727K corpus, consisting of 727,746 English documents for the purpose of training models. The dataset is split into 8:1:1 train:validation:test splits. The supervised baseline models are trained on this dataset.

The test split of the WIKI-727K is used for the evaluation of text segmentation models. However, some models consume a large number of computational resources, for segmenting the dataset. Hence, evaluating them on the test split becomes infeasible. For this reason, the authors also released the WIKI50, consisting of only 50 documents. Models consuming a large number of resources are evaluated on this dataset.

4.2 Elements and Cities

Chen et al. (2009) created two small datasets, *CITIES* and *ELEMENTS*, from Wikipedia based on the cities and elements of the world. These datasets have been used for evaluation of the performance of text segmentation models.

4.3 CHOI

Choi dataset consists of 920 documents, created by concatenation of 10 paragraphs randomly sampled from Brown corpus. The documents are distributed to different folders based on the number of sentences in each paragraph. This dataset has been used for the evaluation of the performance of text segmentation models.

5 Approaches

5.1 Discourse Cue Words

Usually, in certain domains of text, there exist certain words which can be used for detecting the end of a paragraph. These words are called cue words or boundary markers. The presence of the cue words makes the text segmentation task easy.

Eg. Today's breaking news, seven terrorists have been caught in Area X. ***Coming to the next news,*** *an asteroid is approaching Earth at very high speed*
Here **Coming to the next news,** is the discourse cue word.

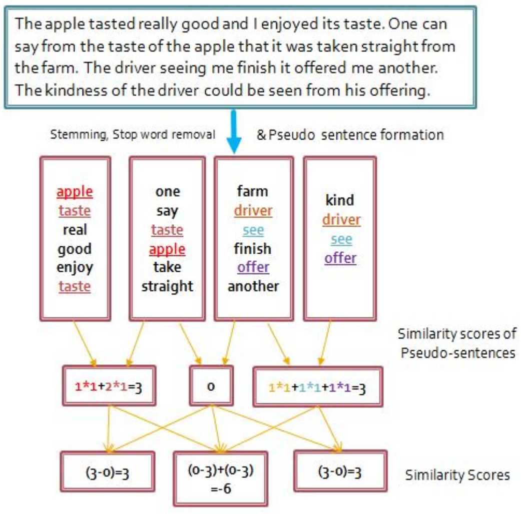
However, the discourse cue word is context-dependent. The keywords usable in a certain domain, e.g., cooking, cannot be used in other domains, e.g., sports.

5.2 Subtopic shift

(Hearst, 1997) uses subtopic shift for the detection of paragraph boundaries in text. The author discusses that the term topic is hard to define and detect and thus uses techniques to detect the change in topic inside text. She uses techniques like new word introduction and change in vocabulary to detect the subtopic shift. The author proposes the approach of **TextTiling**, which uses cohesion to detect the topic shift using lexical similarity. The approach consists of 3 steps:

- Tokenization: The text is converted into lowercase and gets tokenized. Stemming of the words is done, and stop words (like 'a,' 'the,' 'he,' etc.) get removed from the token set. Consecutive words are grouped to form pseudo-sentences, with the preferred length being 20. The pseudo-sentences are grouped to form a vector of size equal to the count of unique words in the discourse. Each position in the vector corresponds to the count of the word, linked with that position, present in the pseudo-sentence with the default value being 0.

- Lexical Score Determination: Cosine Similarity Score between vectors of two consecutive pseudo-sentences is calculated. The score is allocated to the pseudo-sentence gaps.


The apple tasted really good and I enjoyed its taste. One can say from the taste of the apple that it was taken straight from the farm. The driver seeing me finish it offered me another. The kindness of the driver could be seen from his offering.

Stemming, Stop word removal
& Pseudo sentence formation
apple taste real good enjoy taste
one say taste apple take straight
farm driver see finish offer another
kind driver see offer
Similarity scores of Pseudo-sentences
1*1+2*1=3
0
1*1+1*1+1*1=3
(3-0)=3
(0-3)+(0-3) =-6
(3-0)=3
Similarity Scores

Figure 2: Similarity Score Calculation in TextTiling

- Boundary Identification: The depth value at pseudo-sentence gap 'i' is calculated using the formula: $(g_{i-1} - g_i + g_{i+1} - g_i)$, where g_i is the score at gap i. This value for all pseudo-sentences is compared with the average depth value across all pseudo-sentences and if the value turns out to be less than the average depth value, we put a paragraph there.

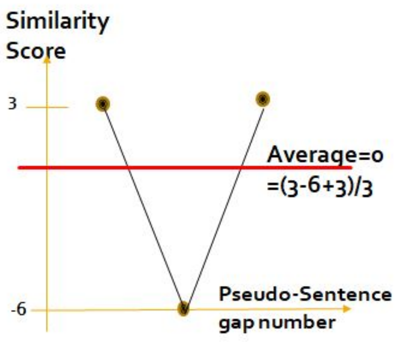
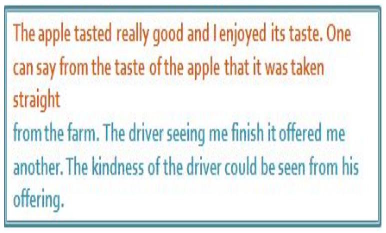


Figure 3: Boundary Scores below threshold


The apple tasted really good and I enjoyed its taste. One can say from the taste of the apple that it was taken straight

from the farm. The driver seeing me finish it offered me another. The kindness of the driver could be seen from his offering.

Figure 4: Output of TextTiling

This approach is extremely intuitive. However, the focus of the approach was majorly on lexical similarity (Cohesion). The authors could have utilized synonym, hyponym, and hypernym based similarity.

5.3 Topic Similarity

Riedl and Biemann (2012) discusses the use of the technique (**TopicTiling**), which deals with topic similarity and detects boundaries based on the degree of similarity.

In the beginning, 'T' numbers of topics are selected. Using the LDA (Latent Dirichlet Allocation) inference approach, each word is assigned a topic, out of 'T' topics. The topic with the highest probability gets assigned to the word. In this way, all of the words are allocated a one-hot coded vector of T-dimensions with 1 in the position corresponding to the topic the word belongs in.

Blocks are created from the sentences, and each block has an associated T dimensional vector where each position indicates the count of words for the corresponding topic. The value of the block can be calculated by adding the vector of all of the words present in the block.

A window size ‘w’ is taken, and for each sentence gap ‘p,’ ‘p-w’ to ‘p+w’ blocks are used to find the cohesion score for that gap. The score is found using cosine similarity between the vectors of the two blocks. When the cosine score falls below a threshold, a boundary is said to be detected between the two consecutive blocks.

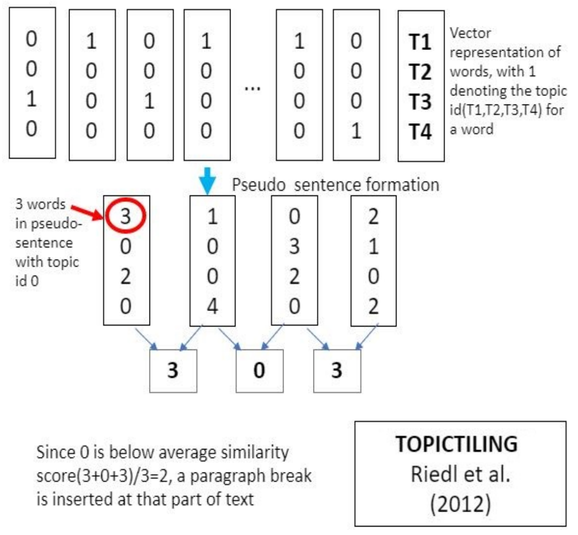

Vector representation of words, with 1 denoting the topic id(T1,T2,T3,T4) for a word
T1
T2
T3
T4
Pseudo sentence formation
3 words in pseudo-sentence with topic id 0
Since 0 is below average similarity score(3+0+3)/3=2, a paragraph break is inserted at that part of text
TOPICTILING
Riedl et al.
(2012)

Figure 5: Example of Topic Tiling

5.4 Graph Based Approach

Glavaš et al. (2016) propose an unsupervised graph-based approach (*GraphSeg*). The approach consists of building a *Semantic relatedness graph* with the sentences comprising the nodes, and the semantically related sentences are connected by edges. Coherent segments are obtained by finding the maximal cliques in the graph.

The approach starts with building a graph with no edges. Only if the semantic similarity of the nodes is above a threshold τ, the nodes get connected by edges. Next, the approach finds the maximal cliques in the graph. Next, the approach creates segments by merging adjacent sentences found in at least one clique. After this step, two adjacent segments are merged if there exists a clique with at least one sentence from each of the segments.

The approach then uses the minimum segment size ‘n’ and finds segments with less than ‘n’ sentences. The segment is merged with its neighboring segment with the maximum semantic similarity. The entire process can be better understood by the example taken from Glavaš et al. (2016), as shown in Figure 6.

Step	Sets
Cliques Q	{1, 2, 6}, {2, 4, 7}, {3, 4, 5}, {1, 8, 9}
Init. seg.	{1, 2}, {3, 4, 5}, {6}, {7} {8, 9}
Merge seg.	{1, 2, 3, 4, 5}, {6}, {7}, {8, 9}
Merge small	{1, 2, 3, 4, 5}, {6, 7}, {8, 9}

Figure 6: Creating segments from graph cliques (n = 2). In the third step we merge segments {1, 2, 3} and {4, 5} because the second clique contains sentences 2 (from the left segment) and 4 (from the right segment). In the final step we merge single sentence segments (assuming segs({1, 2, 3, 4, 5}, {6}) ¡ segs({6}, {7}) and segs({7}, {8, 9}) ¡ segs({6}, {7}))

5.5 Supervised approach

Koshorek et al. (2018) adopts a supervised deep learning approach that uses LSTMs and feed-forward neural networks. The work uses two-layer Bidirectional LSTM or BiLSTM with Max Pooling to find sentence embeddings.

Their model takes a set of sentences as input and uses the BiLSTM architecture to convert each sentence into its sentence embeddings. The sentence embedding for all the input sentences is passed through another two-layered BiLSTM. The output across all time steps of the BiLSTM is passed through a Softmax layer. The output of the softmax layer is used to predict which sentence(s) in the input, demarcate the start of a new paragraph.

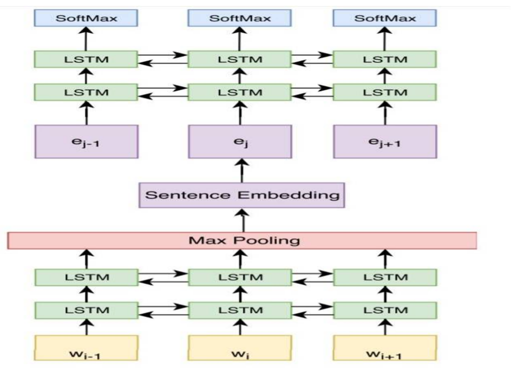


Figure 7: Text Segmentation Model of Koshorek et al. (2018)

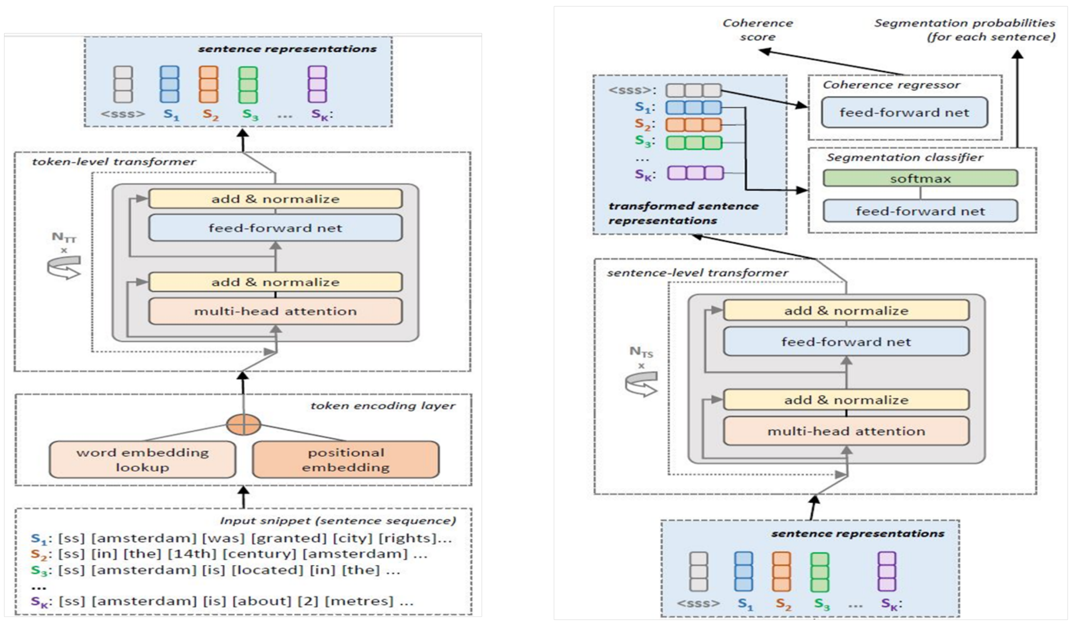


Figure 8: Text Segmentation Model of Glavaš and Somasundaran (2020)

5.6 Supervised approach with auxiliary coherence modeling

Glavaš and Somasundaran (2020) uses the transformer for the purpose of text segmentation. They use transformer encoders for generating sentence embeddings. The sentence encodings are then passed through another transformer encoder, which generates better sentence representations. The sentence representations are then passed through a feed-forward neural network. The softmax output of the feed-forward layer is used to calculate the probability, that whether a sentence is the start of the new paragraph or not.

The authors paired the task of text segmentation with auxiliary coherence modeling. The multitasking model produces better results than their standalone text segmentation model and achieves the **state of the art** results on benchmark text segmentation datasets.

6 Results and Analysis

From Glavaš and Somasundaran (2020), we find the performance of the text segmentation models across some well-established datasets. TLT-TS and CATS are their text segmentation models.

All models have been compared with a Random baseline model. The Random model, predicts a new paragraph break with a probability of

$$\frac{Number\ of\ Paragraphs\ in\ Text}{Number\ of\ Sentences\ in\ Text} \quad (2)$$

The P_k scores for the models across different datasets have been reported in the table. In the experiments, the value of k is set to half of the average reference segment length.

From the table, we find that CATS is achieving the state of the art performance across 4 datasets. Whereas, the unsupervised approach, GraphSeg, has the state of the art performance in the Choi dataset.

Also, it can be seen that Supervised models have better (lower) P_k scores in comparison to unsupervised models. Hence, training models to detect patterns in the text is useful for text segmentation.

Lastly, we see that CATS, which is the model with auxiliary coherence modeling, has the best performance out of all models. It is performing better than its counterpart, which is not using coherence modeling. So pairing the text segmentation task with discourse phenomena tasks (Coherence modeling) significantly improves the performance of the model.

7 Conclusion

Text Segmentation is the task of breaking text into meaningful paragraphs. Text segmentation models cannot be evaluated by metrics like Precision, Recall & F-score, and require special metrics like P_k and WindowDiff. Text Segmentation datasets are available online for training and checking the performance of models. A variety of approaches have been taken to solve the problem of text segmenta-

Model	Model Type	WIKI 727K	WIKI50	Choi	Cities	Elements
Random	Unsupervised	53.09	52.65	49.43	47.14	50.08
GraphSeg	Unsupervised	-	63.56	**5.6-7.2**	39.95	49.12
Koshorek et al. (2018)	Supervised	22.13	18.24	26.26	19.68	41.63
TLT-TS	Supervised	19.41	17.47	23.26	19.21	20.33
CATS	Supervised	**15.95**	**16.53**	18.50	**16.85**	**18.41**

Table 1: Performance of different text segmentation model across standard English text segmentation datasets

tion. The state of the art performance is achieved by Glavaš and Somasundaran (2020) across a majority of datasets, while Glavaš et al. (2016) achieves the state of the art result in Choi dataset.

References

Doug Beeferman, Adam Berger, and John Lafferty. 1999. Statistical models for text segmentation. *Machine learning*, 34(1-3):177–210.

Harr Chen, SRK Branavan, Regina Barzilay, and David R Karger. 2009. Global models of document structure using latent permutations. Association for Computational Linguistics.

Wesley T Chuang and Jihoon Yang. 2000. Extracting sentence segments for text summarization: a machine learning approach. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 152–159.

Gaël Dias, Elsa Alves, and José Gabriel Pereira Lopes. 2007. Topic segmentation algorithms for text summarization and passage retrieval: An exhaustive evaluation. In *AAAI*, volume 7, pages 1334–1340.

Goran Glavaš, Federico Nanni, and Simone Paolo Ponzetto. 2016. Unsupervised text segmentation using semantic relatedness graphs. Association for Computational Linguistics.

Goran Glavaš and Swapna Somasundaran. 2020. Two-level transformer and auxiliary coherence modeling for improved text segmentation. *arXiv preprint arXiv:2001.00891*.

Marti A Hearst. 1997. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational linguistics*, 23(1):33–64.

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

Lev Pevzner and Marti A Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36.

Mohsen Pourvali and Ph D Mohammad Saniee Abadeh. 2012. A new graph based text segmentation using wikipedia for automatic text summarization. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 3(1).

Martin Riedl and Chris Biemann. 2012. Topictiling: a text segmentation algorithm based on lda. In *Proceedings of ACL 2012 Student Research Workshop*, pages 37–42. Association for Computational Linguistics.