A Neural CRF-based Hierarchical Approach for Linear Text Segmentation

Inderjeet Nair, Aparna Garimella, Balaji Vasan Srinivasan, Natwar Modani, Niyati Chhaya, Srikrishna Karanam, Sumit Shekhar
Adobe Research, India
{inair,garimell,balsrini,nmodani,
nchhaya,skaranam,sushekha}@adobe.com

Abstract

We consider the problem of segmenting unformatted text and transcripts linearly based on their topical structure. While prior approaches explicitly train to predict segment boundaries, we propose to address this task by inferring the hierarchical segmentation structure associated with the input text. For this purpose, we present a data curation strategy to obtain the hierarchical segmentation structure annotations for over 700K Wikipedia articles. We then propose the first supervised approach to generate hierarchical segmentation structures for given text based on a neural conditional random field (CRF) that explicitly models the statistical dependencies between nodes and their constituent children. We introduce a novel data augmentation scheme as part of our model training, which involves sampling a variety of node aggregations, permutations, and removals, all of which help capture fine-grained and coarse topical shifts in the data and improve model performance. Extensive experiments show that our model outperforms or achieves competitive performance when compared to previous state-of-the-art algorithms in the following settings: rich-resource, cross-domain transferability, few-shot supervision, and segmentation when topic label annotations are provided.

1 Introduction

Text segmentation (Hearst, 1997; Choi, 2000), an important task in information retrieval, is defined as the process of dividing unstructured text into topically coherent segments. Because it recovers topical structure from unformatted text, it can be used as a pre-processing step for several downstream tasks such as text summarization (Mitra et al., 1997), question answering (Oh et al., 2007) and discourse analysis (Van Dijk, 1982).

Most prior works on text segmentation (Hearst, 1997; Choi, 2000; Koshorek et al., 2018) attempted to address this task by explicitly *predicting the segment boundaries*, with the assumption that any given text can be *decomposed* into contiguous, non-overlapping, indivisible segments, based on topical themes. The discourse segmentation theory (Grosz and Sidner, 1986), however, asserts that the outcome may not always be strictly decompositional, *i.e.*, a segment may have sub-segments within it, and segments may overlap with each other. Following this theory, we hypothesize that explicitly training to infer the hierarchical topic structure of the underlying text leads to better linear segmentation, as it forces the models to examine text at multiple levels to extract coarse-grained to fine-grained topical segments. Further, this allows inference of linear segments of varying granularity that can be used for various downstream applications.

Previous works on hierarchical segmentation are largely unsupervised (Eisenstein, 2009; Simon et al., 2015) due to the unavailability of large labelled datasets with hierarchical structure information. In this paper, we propose to leverage the hierarchical structures in a supervised manner, given the superior performances of supervised models across several language processing tasks (Mikolov et al., 2013; Pennington et al., 2014; Devlin et al., 2019), and propose a data curation strategy to obtain the hierarchical segmentation structures for Wikipedia articles. Specifically, we leverage the available HTML tag annotations[1] and use them to identify section and sub-section information with their hierarchical level, which are then leveraged to obtain the associated ground truth hierarchical structure. Further, because these are extracted from Wikipedia dump, they cover a wide range of topics unlike prior/existing datasets (Eisenstein, 2009).[2]

Our approach is based on a recent CRF-based constituency chart parsing technique (Zhang et al., 2021), which offers efficient algorithms for super-

[1] https://dumps.wikimedia.org/
[2] Note that the dataset proposed by Eisenstein (2009) is a small one consisting of 12 examples for evaluation, and will not suffice for training large models.

vised training and precise inference. This framework explicitly models the relationships between nodes and their offspring in binary trees, and thus can enable hierarchical segmentation inference by utilising the relationships between coarse segments and their fine-grained sub-segments. However, there are three challenges to directly adapt this method to hierarchically segment text: (a) In contrast to the abundant labelled resources available for constituency parsing (Marcus et al., 1993; Xue et al., 2005), there is no large-scale labelled dataset for this task. (b) This method can only infer binarized hierarchical structures; it cannot be extended to infer general hierarchical structures with nodes having any number of children, as training and inference become infeasible. (c) While the existing method processes a sequence of tokens, the input in our case would be a sequence of sentences.

We propose a framework for linear text segmentation using the hierarchical structures of the underlying text. Specifically, our work makes four main contributions: **(1)** We design an algorithm to obtain the hierarchical structures for Wikipedia articles in HTML format, and curate a large labelled dataset for hierarchical text segmentation.[3] **(2)** We present an algorithm based on the Chomsky Normal Form (CNF) (Chomsky, 1959; Hopcroft et al., 2001; Lange and Leiß, 2009) theory to convert the hierarchical structures to binarized form - which makes the computation of the tree-structure CRF objective tractable. **(3)** We propose a Transformer-based architecture (Vaswani et al., 2017) to encode the input sequence's sentences, which uses a lot fewer parameters than previous state-of-the-art BERT-based (Devlin et al., 2019) approaches (Lukasik et al., 2020). **(4)** We further propose a data augmentation technique involving random node aggregations, removals and permutation, which results in significant performance improvement. Finally, we demonstrate our method's efficacy by comparing its performance against prior unsupervised and supervised linear text segmentation approaches.

2 Related Works

Prior works for linear text segmentation can be divided into unsupervised and supervised methods, both of which can be further categorized into locally and globally-informed ones. Locally-informed methods find segment boundaries by estimating the extent of topical shift using local cues (Hearst, 1997; Blei and Moreno, 2001; Lafferty et al., 2001). While these methods enjoy quick inference and low memory constraints as they only utilize local features, they can result in erroneous predictions when met with short inconsequential digressions (Kazantseva and Szpakowicz, 2011). Globally-informed methods, on the other hand, utilize the complete context in optimizing an objective to find the locations of topical shift (Choi, 2000; Kazantseva and Szpakowicz, 2011; Malioutov et al., 2007; Fragkou et al., 2004; Glavaš et al., 2016). As they consider the entire global context in inference, these methods have higher memory constraints and time requirements.

More recently, Koshorek et al. (2018) introduced a large-scale dataset for linear text segmentation, which has resulted in the application of supervised neural models to predict the segment boundaries for unstructured text (Koshorek et al., 2018; Badjatiya et al., 2018; Li et al., 2018). These models not only achieve better performance but also are endowed with high inference speed, owing to parallelized computing with modern GPU architectures.

Owing to the success of supervised methods for linear text segmentation, we design a globally-informed supervised neural model for predicting segment boundaries. However, unlike previous works which address segment boundary prediction explicitly, ours first hierarchically segments the text, and then leverages the resulting structures to predict the linear segment boundaries. To enable the supervised training of our proposed approach, we curate a large labelled dataset consisting of Wikipedia articles along with their hierarchical structures automatically. Further, our proposed method requires significantly fewer parameters that prior SoTA globally-informed methods while achieving better performances. To the best of our knowledge, ours is the first work to leverage hierarchical structures to predict segment boundaries, and show that this results in improved performances for the task of linear text segmentation.

3 Problem Formulation

Here, we briefly outline the objectives of linear and hierarchical text segmentation tasks. Given an article $\mathbf{S}$ composed of n sentences, $\mathbf{S} = s_0, s_1, \ldots, s_{n-1}$, the goal of linear segmentation is to obtain a contiguous partition $L =$

[3]The code to curate dataset is available at https://github.com/inderjeetnair/hierarchical_text_segmentation_data

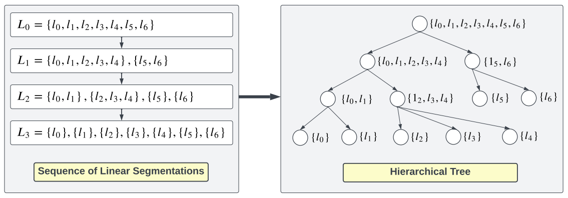
$L_0 = \{l_0, l_1, l_2, l_3, l_4, l_5, l_6\}$
$L_1 = \{l_0, l_1, l_2, l_3, l_4\}, \{l_5, l_6\}$
$L_2 = \{l_0, l_1\}, \{l_2, l_3, l_4\}, \{l_5\}, \{l_6\}$
$L_3 = \{l_0\}, \{l_1\}, \{l_2\}, \{l_3\}, \{l_4\}, \{l_5\}, \{l_6\}$
Sequence of Linear Segmentations
$\{l_0, l_1, l_2, l_3, l_4, l_5, l_6\}$
$\{l_0, l_1, l_2, l_3, l_4\}$
$\{1_5, l_6\}$
$\{l_0, l_1\}$
$\{1_2, l_3, l_4\}$
$\{l_5\}$
$\{l_6\}$
$\{l_0\}$
$\{l_1\}$
$\{l_2\}$
$\{l_3\}$
$\{l_4\}$
Hierarchical Tree

Figure 1: Transformation of a sequence of linear segmentations to a hierarchical tree.

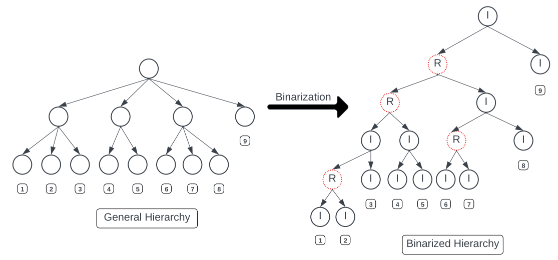

Figure 2: **Binarization:** Transformation of a tree having nodes containing more than 2 children to a binarized form.

$l_0, l_1, \ldots, l_{k-1}$ such that joining the elements of l_i in the same order reconstructs $\mathbf{S}$ and $l_i \cap l_j = \phi \, \forall \, i \neq j$. Each segment l_i in L is associated with a topical theme which can be used for downstream tasks such as summarization, information retrieval, etc.

Hierarchical segmentation (McFee et al., 2017) aims to infer a sequence of linear segmentations, $\mathbf{L} = L_0, L_1, \ldots, L_{m-1}$, where L_i is coarser than L_j for $i < j$. Each element of $\mathbf{L}$ is thus a refinement of all its preceding elements, to satisfy this coarse-to-fine grained constraint. The refinement condition for $i \leq j$ is: $\forall l \in L_j \; \exists \; l' \in L_i : l \subseteq l'$. That is, every segment in L_j is a subset of a segment in L_i. In this paper, we represent the information contained in $\mathbf{L}$ using a hierarchical tree (Fig. 1), where each node (other than the leaf nodes) represents a topical theme. The nodes near the root represent general / coarser topics, and those near the leaves indicate specific / fine-grained topics. In our approach, the inferred hierarchical segmentation is in the form of a tree. After converting this tree to a sequence of linear segmentations, we return an appropriate element from the sequence as our inferred linear segmentation.

4 Dataset Curation

To train our proposed method in a supervised manner, we collect hierarchical segmentation structure annotations for the Wikipedia articles in WIKI-727K dataset (Koshorek et al., 2018). As done in (Koshorek et al., 2018), the articles in the HTML form are preprocessed using WikiExtractor[4] to remove (a) non-text elements such as tables and figures, and (b) very short sections and sub-sections spanning fewer than three sentences. The markup tags (`<h1>, ..., <h6>`) associated with different sections define the level of hierarchy for a given text segment. We leverage this markup information to obtain the hierarchical structure among the various segments. We thus obtain a sequence of HTML elements for each article. In the next sub-section, we describe our algorithm to obtain the hierarchical structures associated with these sequences. We obtain the hierarchical structure annotations only for the train split articles of the WIKI-727K dataset; our approach can be applied to larger document collections to obtain more datapoints.

[4]https://github.com/attardi/wikiextractor.git (Distributed under GNU Affero General Public License v3.0)

4.1 Hierarchical Labelling

Algorithm 1 Algorithm for constructing hierarchical structure from a list of HTML elements

```
Require: $\mathcal{X} = x_1, x_2, \ldots, x_L$          ▷ Ordered list of HTML elements
  $c \leftarrow$ ROOT          ▷ ROOT initialized denoting the root of the tree to be constructed
  $\mathcal{T} \leftarrow$ ROOT
  for $i = 1$ to $\|\mathcal{L}\|$ do
    $x \leftarrow \mathcal{X}[i]$
    while PRIORITY($x$.TAG) $\geq$ PRIORITY($c$.TAG) do          ▷ Selecting appropriate element to add $x$
      $c \leftarrow c$.PARENT          ▷ Updating $c$ to its parent
    end while
    $c$.ADD($x$)          ▷ $x$ is added as the next child of $c$
    $c \leftarrow x$
  end for
  Return $\mathcal{T}$
```

Let the sequence of HTML elements associated with an article be $\mathcal{X} = x_0, x_1, \ldots x_{L-1}$, where x_i.TAG denotes the markup type associated with x_i, and x_i.TEXT denotes its associated text. Here, we outline our algorithm to obtain the hierarchical organization of these elements. Let this hierarchical organization be represented as a tree rooted at $\mathcal{T}$. The non-leaf nodes represent topics, while the leaf nodes represent sentences from the article.

Our algorithm iterates over the elements in $\mathcal{X}$ and progressively adds them to the tree rooted at $\mathcal{T}$. It maintains a reference to the node c that is last added to the tree. To add the next element x, the algorithm only considers two possibilities: (a) x is the next child of c, or (b) x is the next child of one of the ancestors of c. This is to ensure that the pre-order traversal of $\mathcal{T}$ recovers $\mathcal{X}$ (which happens when x is added using the above two rules). To find

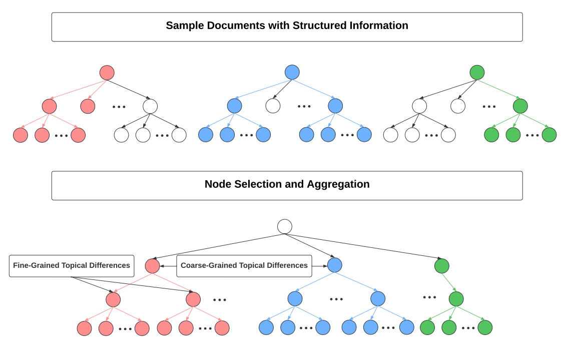
Sample Documents with Structured Information
Node Selection and Aggregation
Fine-Grained Topical Differences
Coarse-Grained Topical Differences

Figure 3: **Data Augmentation:** First, hierarchical structures are randomly sampled from the corpus; then, some nodes are removed from the sampled structures and the outcome is combined

the element to which x must be added, we associate a priority to each markup type in the following decreasing order: `h1, ..., h6, p`, where $\mathtt{h}_i$ indicates for section / sub-section headings and `p` its associated text. For adding x, c is updated to its parent until the priority of c exceeds that of x. Algorithm 1 presents the pseudo code.

4.2 Binarization

Algorithm 2 Algorithm to be applied to every node having more than 2 children to convert the original structure to the binarized form

```
Require: $x$                                   ▷ Node having more than 2 children
  $c \leftarrow$ NEWNODE()
  $c$.TYPE $\leftarrow$ I
  $n \leftarrow ||x$.CHILDREN$||$
  for $i = 1$ to $n$ do
      $c' \leftarrow$ NEWNODE()                    ▷ New node initialized
      $c'$.TYPE $\leftarrow$ R
      $c$.CHILDREN $\leftarrow [c', x$.CHILDREN$[n - i + 1]]$    ▷ Restricting the number of productions to 2
      $c \leftarrow c'$
  end for
```

The hierarchies thus obtained allows nodes to have more than two children. We convert them to binarized form (from which the original structures can be recovered) to ensure tractable training and inference using our CRF-based segmentation model (§5) (using Algorithm 2).

This algorithm visits each node x in a tree $\mathcal{T}$ that has more than two children, and partitions the children into two sets having $||x.\text{CHILDREN}|| - 1$ children and 1 child respectively. Thereafter, a new node is constructed whose children are assigned to the former set, followed by the updation of x.CHILDREN to contain the new node and the latter set in the partition. This is repeated until the tree is devoid of nodes with more than 2 children. To ensure recoverability, we define two types of nodes in the binarized trees: **Reducible (R)** and **Irreducible (I)**. The nodes retained from $\mathcal{T}$ are regarded to as **I**, and those added to convert $\mathcal{T}$ to the binarized form are referred to as **R** (an example of binarization is shown in Fig. 2). These types are assigned to the node's 'TYPE' property (pseudo code in Algorithm 2). To recover the original tree, we visit every **Reducible** node in the binarized tree and connect its children to its parent in the same order. This process is repeated until the structure becomes devoid of any **Reducible** nodes.

4.3 Data Augmentation

An inherent limitation of this dataset stems from the fact that each Wikipedia page is composed of a single global topic, and the direct usage of this data will only train the model in detecting fine-grained topical shifts resulting from sub-sections / sub-headings. However, an article in practice can also contain fragments with stark topical contrast.

To overcome this, we introduce a data augmentation strategy, where a subset of tree root references are sampled at every iteration. Thereafter, some of the children of these nodes are randomly dropped and the ordering of left-out children is randomly permuted. Finally, a new node is created and its children are the sampled tree roots (Figure 3). This new root consists of several coarse topics and the random permutation of the child nodes ensures that the model robustly infers topical segments independent of the order of the child nodes. The augmentation is performed at every epoch ensuring the number of artificially synthesized datapoints is equal to the actual number of documents in the train split.

5 Neural CRF Segmentation Model

5.1 CRF Formulation

We consider an article containing n sentences, $\mathbf{S} = s_0, s_1, \ldots, s_{n-1}$ and its corresponding hierarchical segmentation tree structure $\mathbf{t}$. A node in $\mathbf{t}$ representing a segment spanning $s_i, s_{i+1}, \ldots, s_j$ is denoted by (i, j). Alternatively, $\mathbf{t}$ can be expressed as a set of tuples where each tuple corresponds to a node segment in $\mathbf{t}$.

Inspired by (Zhang et al., 2021), our model presents a scoring function $s(.,.) \rightarrow \mathbb{R}$ (described later) to assign a score for each node in $\mathbf{t}$, *e.g.*, $s(i, j)$ represents the score for a node entailing $s_i, s_{i+1}, \ldots, s_j$. We define a function $\mathcal{S}$ to score the tree $\mathbf{t}$ using the sequence $\mathbf{S}$ as:

$$\mathcal{S}(\mathbf{S}, \mathbf{t}) = \sum_{(i,j) \in \mathbf{t}} s(i, j) \tag{1}$$

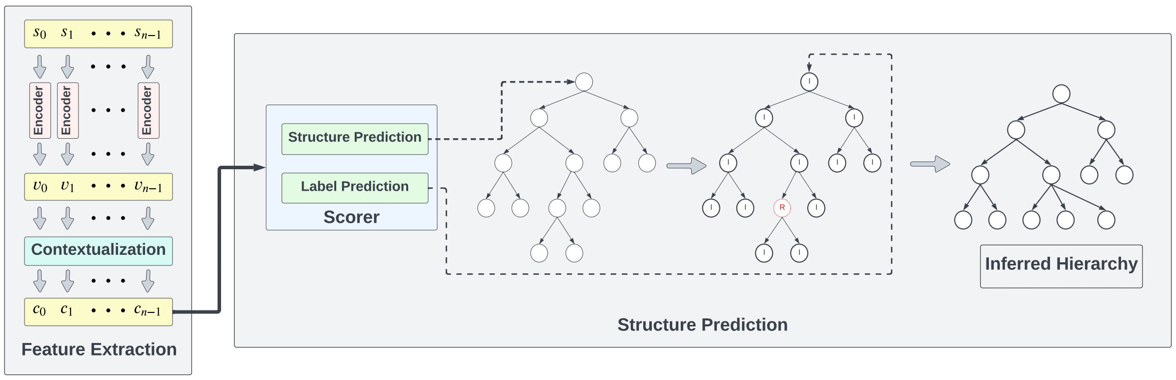

Figure 4: **Pipeline:** Our pipeline is divided into two stages. We extract contextualised embeddings for each sentence during the feature extraction stage. These representations are then fed into the structure prediction module, which first infers the binarized tree and label for each node. This inference is used to construct the actual hierarchy.

Under CRF, we define conditional probability as:

$$\mathbb{P}(\mathbf{t}|\mathbf{S}) = \frac{\mathcal{S}(\mathbf{S}, \mathbf{t})}{Z(\mathbf{S}) \coloneqq \sum_{\hat{\mathbf{t}}} \mathcal{S}(\mathbf{S}, \hat{\mathbf{t}})} \tag{2}$$

The denominator sums the score of all possible legal hierarchical trees.[5] Note that, the computation of the partition function $Z(\mathbf{S})$ in the denominator is intractable (exponential time complexity) if we consider all possible hierarchical trees with nodes having arbitrary number of children. However, binarization offers efficient dynamic programming algorithm to compute the partition function with polynomial time complexity. Thus, we binarize $\mathbf{t}$ to obtain $\tilde{\mathbf{t}}$. However, the binarization also associates a type to each node from the set $\{\mathbf{R}, \mathbf{I}\}$. Under this formulation, every node in $\tilde{\mathbf{t}}$ can be represented by a triplet (i, j, l) which indicates that the corresponding node of type $l \in \{\mathbf{R}, \mathbf{I}\}$ spans $s_i, s_{i+1}, \dots, s_j$. Similar to Zhang et al. (2021)'s two-staged method, we first identify the binarized tree structure that maximises $\mathbb{P}(\tilde{\mathbf{t}}|\mathbf{S})$, whose denominator only adds the scores of the binarized trees, and then determine the type for each of its constituent nodes. To find the optimal structure maximizing $\mathbb{P}(\tilde{\mathbf{t}}|\mathbf{S})$, we leverage Cocke–Younger–Kasami algorithm (CYK) algorithm (Sakai, 1961). For each span (i, j) in the inferred structure, we predict its type l:

$$l = \underset{\hat{l} \in \{\mathbf{R}, \mathbf{I}\}}{\arg\max}\, s(i, j, \hat{l}) \tag{3}$$

Figure 4 shows how our model processes the input sequence to infer the hierarchical segmentation structure. In the next subsection, we specify the architectural details of its components.

[5] Legal hierarchical trees are expected to satisfy two conditions: (1) There should be one-to-one correspondence between the leaf nodes and the constituent sentences. (2) Every node in the tree must span consecutive sequence of sentences.

5.2 Model Architecture

We now describe the implemented architecture, which is adapted from the model proposed by Stern et al. (2017) and Zhang et al. (2021) with two important modifications: (a) utilization of memory-efficient Transformer (Vaswani et al., 2017) model for encoding the sentence in place of word encoder, and (b) better choice of hyper-parameters for hierarchical text segmentation.

Encoder. Each sentence in $\mathbf{S}$ is encoded in a context-independent manner using the Transformer-based model proposed by Wang et al. (2022), which contains 6 layers and 22M parameters. The parameters of this model are fine-tuned using self-attention distillation (Wang et al., 2022) for the compression of large language models like RoBERTa-Large (Liu et al., 2019). This stage transforms $s_0, s_1, \dots s_{n-1}$ to $v_0, v_1, \dots v_{n-1}$ with 384 length each.

Contextualization. To contextualize $v_0, \dots v_{n-1}$, we implement two BiLSTM layers over it. While the architecture implemented by Zhang et al. (2021) comprises of three BiLSTM layers, we observe that direct usage of the same hyper-parameter settings lead to sub-optimal results on a validation set. The final context-aware representation for a sentence is obtained by concatenating the corresponding forward and backward vectors from the last layer. Let these vectors be represented by $c_0, c_1, \dots c_{n-1}$.

Scoring. Having obtained the contextualized representations of the elements in $\mathbf{S}$, we describe the architecture used to compute $s(i, j)$ and $s(i, j, l)$. For the computation of $s(i, j)$, the contextualized representations are passed to two multi-layer perceptron (MLP) modules to obtain the left and right boundary representation vectors (Zhang et al., 2021):

$$r_i^s; l_i^s = \text{MLP}_r^s(c_i); \text{MLP}_l^s(c_i) \tag{4}$$

Similarly, additional set of boundary vectors are derived to compute $s(i, j, l)$ for label prediction:

$$r_i^l; l_i^l = \text{MLP}_r^l(c_i); \text{MLP}_l^l(c_i) \tag{5}$$

The dimension of the boundary vectors for structure prediction is set to 500 and that for label prediction is set to 800. $s(i, j)$ is computed by introducing a trainable parameter $\mathbf{W} \in \mathbb{R}^{d \times d}$:

$$s(i, j) = l_i^{sT} \mathbf{W} r_i^s \tag{6}$$

Similarly $s(i, j, l)$ is computed by introducing $\mathbf{W_R}$ and $\mathbf{W_I}$ to derive the scores: $s(i, j, \mathbf{R})$ and $s(i, j, \mathbf{I})$ respectively. Note we use r_i^l and l_i^l for the computation of these scores of instead of r_i^s and l_i^s.

5.3 Training

An instance in the labelled dataset can be represented by: $(\mathbf{S}, \tilde{\mathbf{t}}, \mathbf{l})$ where $\mathbf{l}$ is the set of all spans annotated with their corresponding type from $\{\mathbf{R}, \mathbf{I}\}$. The loss function is formed by accumulating two components:

$$\mathcal{L}(\mathbf{S}, \tilde{\mathbf{t}}, \mathbf{l}) = \mathcal{L}_s(\mathbf{S}, \tilde{\mathbf{t}}) + \mathcal{L}_l(\mathbf{S}, \tilde{\mathbf{t}}, \mathbf{l}) \tag{7}$$

The first term tries to maximize $\log(\mathbb{P}(\tilde{\mathbf{t}}|\mathbf{S}))$ by refining the scoring function $s(i, j)$. The second term establishes cross-entropy loss for the type prediction of the constituent spans. While the time complexity of the partition function computation $Z(\mathbf{S})$ for $\log(\mathbb{P}(\tilde{\mathbf{t}}|\mathbf{S}))$ is $\mathcal{O}(n^3)$ using inside algorithm (Lari and Young, 1990), we implement the batchified version of this algorithm proposed by Zhang et al. (2021) that provides much better time complexity ($\mathcal{O}(n)$ for a batch).

We train our models over the curated dataset for 4 epochs using Adam optimizer (Kingma and Ba, 2015) with batch size of 100 and learning rate initialized to 2×10^{-4}. The learning rate is exponentially decayed to 0.75 times its initial value after 50K optimizer steps. The training is restricted to datapoints having less than 200 sentences due to GPU memory constraints.

6 Experiments and Results

We assess our method's performance in various settings when compared to SoTA linear segmentation techniques. The tree inference from the model is converted to a sequence of linear segmentations (Fig. 1). As the position of an element in this sequence indicates the extent of segmentation, we select an appropriate position (constant for a dataset, obtained through validation for supervised methods, and second position for unsupervised methods due to how coarsely grained the topical shifts are in them) and return the corresponding segmentation. We call our method variants **HIERCRF** and **HIERCRF-AUG**, where the former and latter are trained without and with data augmentation.

Method	Precision	Recall	F1 Score
BI-LSTM	69.3	49.5	57.7
CROSS SEGMENT BERT	69.1	63.2	66.0
BERT+BI-LSTM	67.3	53.9	59.9
HIERARCHICAL BERT	69.8	63.5	66.5
HIERCRF	80.6	59.4	68.4
HIERCRF-AUG	**82.5**	60.3	69.7
HIERCRF-BERT	79.0	63.3	70.2
HIERCRF-AUG-BERT	80.4	**64.6**	**71.6**

Table 1: Comparison with supervised baselines when abundant labelled data is available. Only the hierarchical structures of the articles in the train split of WIKI-727K are used in our method for consistency.

We compare our models to supervised methods trained on WIKI-727K augmented with our hierarchical structure annotations (§6.1). Unsupervised methods (Kazantseva and Szpakowicz, 2011; Du et al., 2013) require a significant amount of time for inference as they are computationally intensive and do not take advantage of GPU parallelization for efficiency. Hence, we do not compare their performance for WIKI-727K, which contains a large number of datapoints in the test split. In §6.2, we look at how our model transfers knowledge from one domain to another and investigate efficacy in a low-resource setting. Here, we compare our method to unsupervised as well as some supervised methods. Finally, we evaluate how our model utilises topic label information for segmentation using WikiSection (Arnold et al., 2019).

6.1 Rich Resource Setting

Here, we consider models that perform well in linear text segmentation when large labelled datasets are available for training. The large-scaled dataset (WIKI-727K) curated by Koshorek et al. (2018) for linear segmentation has been instrumental in the formulation of several deep learning methods (Koshorek et al., 2018; Lukasik et al., 2020). We use the following as baselines: **BI-LSTM** (Koshorek et al., 2018), **CROSS SEGMENT BERT** (Lukasik et al., 2020), **BERT+BI-LSTM** (Lukasik et al., 2020) and **HIERARCHI-**

Property		Clinical	Fiction	Wiki
# Documents		227	85	300
# Sentences		31,868	27551	58,071
Segment Length	Mean	35.72	24.15	25.97
	Std Dev	29.37	18.24	9.98

Table 2: Datasets used for comparing our method against statistical methods for linear text segmentation.

CAL BERT (Lukasik et al., 2020). Most of them use BERT as their encoder ($>$ 109M parameters). To show the increased effectiveness of our model when its complexity is increased, we also present the performance with BERT as its encoder. We use the test split in WIKI-727K ($73,233$ instances) and F1 score for evaluating the segment boundary prediction performance. Further, the first and the last sentences are not annotated in the ground truth set of boundaries, as any segmentation algorithm can easily predict them as segmentation boundaries, thus inflating performance.

We note, from Table 1, that our linguistically motivated approach for inferring linear segmentation from hierarchical segmentation gives better performance despite having significantly fewer parameters ($\approx$ 23M as opposed to strongest baseline's $\approx$ 109M). As expected, increasing model complexity improves performance, resulting in a new SoTA for WIKI-727K (71.6 F1). We also observe our precision is comparatively higher and the recall is lower. We attribute this to node misclassification in the inferred hierarchy. By construction, **R** nodes are removed to get the final inference indicating the corresponding segments would be absent in the inferred hierarchy. Thus, even if the inferred binarized structure is accurate, a misclassification of **I** nodes as **R** will result in lower recall.

6.2 Cross Domain and Low Resource Setting

We categorize the methods in following groups.

A: Unsupervised: This group comprises of the following unsupervised techniques: **U&I** (Utiyama and Isahara, 2001), **MINCUT** (Malioutov and Barzilay, 2006), **BAYESSEG** (Eisenstein and Barzilay, 2008), **APS** (Kazantseva and Szpakowicz, 2011), **PLDA** (Purver et al., 2006) and **TSM** (Du et al., 2013). These methods use the **number of gold standard segments** and **test data corpus for tuning the hyperparameters**.

B: Cross-Domain Transferability: Here, we pretrain supervised models using WIKI-727K's train split and evaluate on other datasets completely unsupervised, without knowing the number of gold-standard segments. We consider top baselines

Method	Clinical		Fiction		Wiki	
	WD	P_k	WD	P_k	WD	P_k
GROUP A						
U&I	37.6	37.0	45.9	45.9	**36.8**	**36.8**
MINCUT	38.2	36.8	40.5	37.1	38.9	36.4
BAYESSEG	35.3	33.9	**33.7**	**27.8**	39.0	35.9
APS	39.9	39.6	48.0	45.1	38.0	39.2
PLDA	37.3	32.4	43.0	36.1	-	-
TSM	**34.5**	**30.6**	40.8	32.5	-	-
RANDOM	45.9	44.1	51.0	47.5	48.6	48.0
GROUP B						
CROSS SEG BERT	40.8	39.4	44.4	42.7	37.1	36.3
HIER BERT	34.8	33.9	**41.1**	**39.0**	35.6	34.5
HIERCRF	34.4	33.9	43.1	42.4	33.4	30.0
HIERCRF-AUG	**33.7**	**33.0**	42.8	42.2	**30.9**	**28.6**
GROUP C						
CROSS SEG BERT-NO-PT	38.4	35.0	39.4	29.5	40.6	38.0
CROSS SEG BERT	31.0	29.8	34.4	27.6	32.4	27.5
HIER BERT-NO-PT	33.4	32.4	37.8	34.5	39.1	38.1
HIER BERT	38.5	35.2	34.0	**25.5**	35.0	29.1
HIERCRF-NO-PT	33.3	32.2	34.7	34.5	37.0	35.9
HIERCRF	26.7	25.5	33.3	29.9	28.6	26.3
HIERCRF-AUG	**25.2**	**24.4**	**32.6**	28.4	**27.9**	**25.7**

Table 3: **Performance of our model against unsupervised approaches.** All results are averaged for 5 random splits. TSM and PLDA implementations are unavailable to report their performance on Wiki. PT: Pre-training.

from §6.1 (CROSS SEGMENT BERT and HIERARCHICAL BERT) and our variants (HIERCRF and HIERCRF-AUG). We re-implement the supervised baselines as original code is unavailable.

C: Low Resource Setting: Here, we expose the models to 20% of the dataset for supervised learning and evaluate it on the rest of the dataset (results averaged for 5 random seeds). For fine-tuning, our model parameters are optimized to predict the flat-hierarchy associated with the datapoints in the training split. We also report the performance of the model (appended with NO-PRETRAINING) which is not pretrained over WIKI-727K.

We use the following three datasets to compare our method against the statistical unsupervised approaches (Table 2):

Clinical (Eisenstein and Barzilay, 2008). Every document is a chapter from a medical textbook where labeled boundaries represent section breaks.

Fiction (Kazantseva and Szpakowicz, 2011). Each document is a fiction from Project Gutenberg where boundary annotations denote chapter breaks.

Wiki (Badjatiya et al., 2018). 300 articles are randomly sampled from the wikipedia dump where section tag labels are used to annotate boundaries.

We assess the performance of these methods using P_k (Beeferman et al., 1999) and WinDiff (WD) (Pevzner and Hearst, 2002) error metrics. P_k computes the probability that two segments sampled from a document are incorrectly identified as

Method	P_k(City)	P_k(Disease)
SECTOR	14.4	26.3
S-LSTM	9.1	20.0
TRANSFORMER^2	8.2	**18.8**
HIERARCHICAL BERT	8.6	22.4
CROSS SEGMENT BERT	10.1	21.8
HIERCRF*	8.8	20.4
HIERCRF-AUG*	8.5	21.2
HIERCRF	8.1	20.3
HIERCRF-AUG	**8.0**	20.0

Table 4: Comparison with various approaches in leveraging segment labels. * indicates those methods do not use segment labels. For our AUG-appended methods, we augment the flat hierarchies associated with the training datapoints.

belonging to the same segment. WD moves a sliding window across the document and counts the number of instances where the hypothesized and reference segment boundaries are different.

The results shown in Table 3 demonstrate the competitive performance exhibited by our model without any additional fine-tuning (**B**). As our models in **B** are not subjected to separate hyper-parameter tuning for different datasets, our proposed models can be applied to other domains with minimal changes (only the position from the sequence of segmentations inferred from the model needs to be specified). The performance of HIERCRF-AUG in **B** is better than all the methods in **A** (despite not knowing the number of gold-standard segments and tuning hyperparameters over the testing corpus) for Wiki and Clinical dataset which demonstrates the effectiveness of our approach in transferring the knowledge from one domain to another. Fine-tuning our models with small number of datapoints (**C**) provides competitive results for most of the datasets. As expected, the performance in few-shot supervision setting is boosted if the model is pre-trained over WIKI-727K for the supervised approaches. Because the datapoints in the Fiction dataset are longer than in the other datasets, our model performs poorly (§8).

6.3 Results using Segment Labels

We compare our model's segmentation performance to baselines when segment-wise topic labels are given. We use WikiSection's split (Arnold et al., 2019) that comprises of English documents from two domains: diseases (3.6K documents) and cities (19.5K documents). We use 70/20/10 splits for train/dev/test. We assess how well our model uses the segment labels compared to earlier methods. We consider the following approaches: **SECTOR** (Arnold et al., 2019), **S-LSTM** (Barrow et al., 2020) **TRANSFORMER^2** (Lo et al., 2021), **HIERARCHICAL BERT** and **CROSS SEGMENT BERT**. For fair comparison, no model is pretrained on WIKI-727K.

To incorporate the topic label information in training, we use the boundary vectors for label prediction and a scoring mechanism similar to Eqn 6. Specifically, the likelihood that span (i, j) corresponds to topic label $\mathbf{T}$ is proportional to $s(i,j) = {l_i^l}^T \mathbf{W_T} r_i^l$. The parameters $\mathbf{W_T}$ for each topic label $\mathbf{T}$ are trained using cross-entropy loss similar to Eqn 7.

Table 4 shows that after incorporating topic label information, our model provides SoTA performance for the City Domain and competitive performance for the Disease Domain. This suggests that providing auxiliary information, such as topic label information, improves the performance of our model. We believe that using a medical domain specific **encoder** (Gu et al., 2021) would improve our model's performance in the Disease domain.

6.4 Ablation: Training Size and Performance

Here, we investigate the effect of training our models on data splits comprising of articles with varying sizes (number of sentences). Our objective is to demonstrate that the performance of the model can be improved by including datapoints with larger size. However, the limitations in the GPU hardware, has restricted the maximum datapoint size in the training split to 200. We consider three variants of our curated dataset for training: (a) datapoint sizes ≤ 50, (b) datapoint sizes ≤ 100, and (c) datapoint sizes ≤ 200. We report the performances of models trained over these variants on splits containing datapoints with size in the following ranges: $[1, 50]$, $(50, 100]$, $(100, 150]$, $(150, 200]$, $(200, \infty)$, and $[1, \infty)$) from the WIKI-727K test data.

Table 5 shows the F1 scores. We note three main trends: (1) The model performance decreases as the size of the datapoint increases. This can be attributed to the following two reasons. Firstly, majority of the datapoints in our curated dataset have size in the range $[1, 50]$, and the number of datapoints decreases as the range varies from $(50, 100]$ to $(200, \infty)$. Secondly, the ability of the model to produce discriminative contextualized sentence features is dependant on the size of the input datapoint. As the contextualization is brought about by Bi-LSTM module, very long sequences result in vanishing gradient problem, and this results in less effective modeling for longer sequences. (2)

Method	Training Dataset Size Upper-bound	Testing Dataset Size Range					
		$[1, 50]$	$(50, 100]$	$(100, 150]$	$(150, 200]$	$(200, \infty)$	$[1, \infty]$
HIERCRF	50	70.4	57.5	46.6	39.5	30.7	64.2
	100	72.5	62.2	52.3	47.0	39.4	67.4
	200	**72.9**	**64.2**	**54.8**	**50.1**	**41.8**	**68.4**
HIERCRF-AUG	50	71.8	61.3	51.2	45.4	37.2	66.5
	100	73.9	64.1	54.9	50.3	42.5	69.0
	200	**74.2**	**64.2**	**55.2**	50.3	**42.9**	**69.7**

Table 5: **Effect of including datapoints with larger size in the training set:** A model's performance in terms of F_1-Score decreases as the size of the datapoint increases. Increasing the upper bound of the datapoint size in the training dataset improves the performance uniformly for all the dataset size ranges.

Including datapoints with larger size uniformly improves the performance of the model across all the ranges. Thus, one line of future work could be to procure more labels for articles with more number of sentences. However, it is to be noted that training the model over larger sequences imposes heavy requirements on GPU memory. (3) The effectiveness of our data augmentation can be seen here as well in producing uniformly better results than the model not trained over augmented corpus.

7 Conclusion

We curated a dataset with hierarchical structures and introduced an approach for linear segmentation by inferring the hierarchies. We illustrated the effectiveness of our approach against prior supervised and unsupervised methods for several datasets. Our method, when exposed to a small fraction of the data for fine-tuning, achieves superior performance when evaluated on other datasets for this task. Unlike prior unsupervised methods, ours without any hyperparameter tuning achieved competitive results. While we focussed on predicting segment boundaries, our method could also be applied to yield hierarchical segmentation. However, the challenges specific to dataset size and memory persist. We will study these aspects in our future work.

8 Limitations

The ablation studies conducted in this paper highlight some of the limitations of our model. The model predictions are erroneous when the length of the input text is very large. Even if we are able to curate sufficient number of very long text with ground truth hierarchical structure, training the model parameters imposes heavy requirements for GPU memory. This demands for the requirement of a better architecture that not only handles long range sequence dependencies but also is GPU memory-efficient while training.

References

Sebastian Arnold, Rudolf Schneider, Philippe Cudré-Mauroux, Felix A. Gers, and Alexander Löser. 2019. SECTOR: A Neural Model for Coherent Topic Segmentation and Classification. *Transactions of the Association for Computational Linguistics*, 7:169–184.

Pinkesh Badjatiya, Litton J. Kurisinkel, Manish Gupta, and Vasudeva Varma. 2018. Attention-based neural text segmentation. In *Advances in Information Retrieval*, pages 180–193, Cham. Springer International Publishing.

Joe Barrow, Rajiv Jain, Vlad Morariu, Varun Manjunatha, Douglas Oard, and Philip Resnik. 2020. A joint model for document segmentation and segment labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 313–322, Online. Association for Computational Linguistics.

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

David M. Blei and Pedro J. Moreno. 2001. Topic segmentation with an aspect hidden markov model. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, page 343–348, New York, NY, USA. Association for Computing Machinery.

Freddy Y. Y. Choi. 2000. Advances in domain independent linear text segmentation. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

Noam Chomsky. 1959. On certain formal properties of grammars. *Information and Control*, 2(2):137–167.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Lan Du, Wray Buntine, and Mark Johnson. 2013. Topic segmentation with a structured topic model. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 190–200, Atlanta, Georgia. Association for Computational Linguistics.

Jacob Eisenstein. 2009. Hierarchical text segmentation from multi-scale lexical cohesion. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 353–361, Boulder, Colorado. Association for Computational Linguistics.

Jacob Eisenstein and Regina Barzilay. 2008. Bayesian unsupervised topic segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, page 334–343, USA. Association for Computational Linguistics.

Pavlina Fragkou, Vassilios Petridis, and Ath Kehagias. 2004. A dynamic programming algorithm for linear text segmentation. *Journal of Intelligent Information Systems*, 23(2):179–197.

Goran Glavaš, Federico Nanni, and Simone Paolo Ponzetto. 2016. Unsupervised text segmentation using semantic relatedness graphs. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 125–130, Berlin, Germany. Association for Computational Linguistics.

Barbara J. Grosz and Candace L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Comput. Linguist.*, 12(3):175–204.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2021. Domain-specific language model pretraining for biomedical natural language processing. *ACM Trans. Comput. Healthcare*, 3(1).

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

John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. 2001. Introduction to automata theory, languages, and computation. *Acm Sigact News*, 32(1):60–65.

Anna Kazantseva and Stan Szpakowicz. 2011. Linear text segmentation using affinity propagation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 284–293, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Omri Koshorek, Adir Cohen, Noam Mor, Michael Rotman, and Jonathan Berant. 2018. Text segmentation as a supervised learning task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 469–473, New Orleans, Louisiana. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Martin Lange and Hans Leiß. 2009. To cnf or not to cnf? an efficient yet presentable version of the cyk algorithm. *Informatica Didactica*, 8(2009):1–21.

K. Lari and S.J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech Language*, 4(1):35–56.

Jing Li, Aixin Sun, and Shafiq Joty. 2018. Segbot: A generic neural text segmentation model with pointer network. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4166–4172. International Joint Conferences on Artificial Intelligence Organization.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Kelvin Lo, Yuan Jin, Weicong Tan, Ming Liu, Lan Du, and Wray Buntine. 2021. Transformer over pre-trained transformer for neural text segmentation with enhanced topic coherence. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3334–3340, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Michal Lukasik, Boris Dadachev, Kishore Papineni, and Gonçalo Simões. 2020. Text segmentation by cross segment attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4707–4716, Online. Association for Computational Linguistics.

Igor Malioutov and Regina Barzilay. 2006. Minimum cut model for spoken lecture segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 25–32, Sydney, Australia. Association for Computational Linguistics.

Igor Malioutov, Alex Park, Regina Barzilay, and James Glass. 2007. Making sense of sound: Unsupervised topic segmentation over acoustic input. In *Proceedings of the 45th Annual Meeting of the Association of*

Computational Linguistics, pages 504–511, Prague, Czech Republic. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Brian McFee, Oriol Nieto, Morwaread M. Farbood, and Juan Pablo Bello. 2017. Evaluating hierarchical structure in music annotations. *Frontiers in Psychology*, 8.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

Mandar Mitra, Amit Singhal, and Chris Buckley. 1997. Automatic text summarization by paragraph extraction. In *Intelligent Scalable Text Summarization*.

Hyo-Jung Oh, Sung-Hyon Myaeng, and Myung-Gil Jang. 2007. Semantic passage segmentation based on sentence topics for question answering. *Inf. Sci.*, 177:3696–3717.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

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

Matthew Purver, Konrad P. Körding, Thomas L. Griffiths, and Joshua B. Tenenbaum. 2006. Unsupervised topic modelling for multi-party spoken discourse. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 17–24, Sydney, Australia. Association for Computational Linguistics.

Itiroo Sakai. 1961. Syntax in universal translation. In *Proceedings of the International Conference on Machine Translation and Applied Language Analysis*.

Anca Simon, Pascale Sébillot, and Guillaume Gravier. 2015. Hierarchical topic structuring: from dense segmentation to topically focused fragments via burst analysis. In *Recent Advances on Natural Language Processing*.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Masao Utiyama and Hitoshi Isahara. 2001. A statistical model for domain-independent text segmentation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 499–506, Toulouse, France. Association for Computational Linguistics.

Teun A Van Dijk. 1982. Episodes as units of discourse analysis. *Analyzing discourse: Text and talk*, pages 177–195.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2022. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Nat. Lang. Eng.*, 11(2):207–238.

Yu Zhang, Houquan Zhou, and Zhenghua Li. 2021. Fast and accurate neural crf constituency parsing. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20.