# Domain Adaptive Neural Sentence Compression by Tree Cutting

by

Litton J Kurisinkel, Yue zhang, Vasudeva Varma

in

*ECIR*

Report No: IIIT/TR/2019/-1

Centre for Language Technologies Research Centre
International Institute of Information Technology
Hyderabad - 500 032, INDIA
February 2019

# Domain Adaptive Neural Sentence Compression by Tree Cutting

Litton J Kurisinkel[1], Yue Zhang[2], and Vasudeva Varma[3]

[1] International Institute of Information Technology, Hyderabad, India, 500031
`litton.jKurisinkel@research.iiit.ac.in`
[2] Westlake Institute for Advanced Study, West Lake University, Hangzhou, Zhejiang Province, China
`yue.zhang@wias.org.cn`
[3] International Institute of Information Technology, Hyderabad, India, 500031
`vv@iiit.ac.in`

**Abstract.** Sentence compression has traditionally been tackled as syntactic tree pruning, where rules and statistical features are defined for pruning less relevant words. Recent years have witnessed the rise of neural models without leveraging syntax trees, learning sentence representations automatically and pruning words from such representations. We investigate syntax tree based noise pruning methods for neural sentence compression. Our method identifies the most informative regions in a syntactic dependency tree by self attention over context nodes and maximum density subtree extraction. Empirical results show that the model outperforms the state-of-the-art methods in terms of both accuracy and F1-measure. The model also yields a comparable accuracy in readability and informativeness as assessed by human evaluators.

**Keywords:** Sentence Summarization · Sentence Compression · Syntactic Tree Pruning.

## 1 Introduction

Sentence simplification is the task of deriving a noise-free condensation that holds the most abstract information from a noisy complex sentence. The task is useful for solving NLP problems such as summarization. Significant amount of research has been done on the task. Extractive methods [3, 8, 12] remove the noisy portions of a sentence while leaving the result grammatically correct (Figure 1). Abstractive techniques [2] construct a semantic representation for the original sentence and generates the most informative content in its own learnt writing style.

Among extractive method for sentence compression, traditional method leverage statistical m models by exploiting useful manual features to prune noisy regions of a syntactic tree [5]. Recently, neural network has been used for the task [18, 2, 6]. The basic idea is to leverage a recurrent neural network to automatically extract syntactic and semantic features from a sentence, before classifying

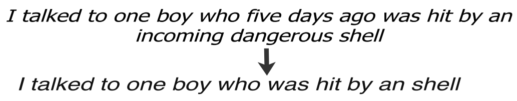
I talked to one boy who five days ago was hit by an incoming dangerous shell
I talked to one boy who was hit by an shell

**Fig. 1.** Sentence Compression

whether each word can be removed. Such methods have shown highly competitive accuracies compared with traditional statistical techniques, without the need to define manual features.

On the other hand, neural methods without syntax typically require a large amount of training data to avoid overfitting, and can generalize poorly across domains. For example, [2] trained their model on Gigaword [15], which consists of 3.8 million article-title pairs. Apparently, datasets on such scales may not be available across domains. [23] show that syntactic information can be useful for alleviating cross-domain performance loss by adding embeddings of syntactic category information to input word representations. This demonstrates that syntax as a layer of abstract structural information can be useful for reducing domain variance. The method of [23] gives the current state-of-the-art accuracies on the task. However, unlike traditional syntax-based statistical methods, their method does not explicitly leverage tree structural knowledge.

We try to combine the potential strengths of syntactic tree noise-pruning approaches and neural models to identify the informative regions in a syntactic dependency tree for domain adaptive sentence simplification. Our method computes the probability of each node to be retained for a simplification process by self attention over context. In addition, we leverage maximum density subtree extraction algorithm to trace out informative regions in a syntactic dependency tree without disturbing grammaticality.

On standard benchmark datasets, our method yields state-of-the-art result measured in terms of both F1-measure and accuracy. To the best of our knowledge, we are the first to investigate tree structures for neural sentence compression. Our code will be released.

## 2 Problem Definition

We solve the problem of sentence compression as a syntactic dependency tree node selection task. Formally, denote a dependency tree $T$ as

$$T \leftarrow \{N, E\}, \tag{1}$$

where $N$ is a set of nodes containing words $w_1, w_2, w_3, .., w_n$ of the sentence and $E$ is the set of dependency edges of $T$. The compressed sentence S is represented by a set of binary labels $\{y_1, y_2, y_3, .., y_n\}$.

$$y_i = \begin{cases} 1 \text{ word at the node } w_i \text{ is retained} \\ 0 \text{ otherwise} \end{cases}$$

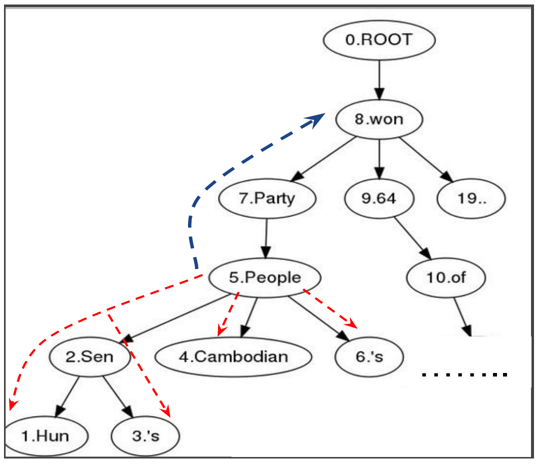

**Fig. 2.** Marking context nodes for the node 'People'

We aim to create a model which can predict the label of a node word using the syntactic context within a dependency tree. By leveraging syntax for label prediction, we expect the model to be generalizable across domains. We train our model using an annotated dataset containing a set of parallel records consisting of dependency trees and corresponding deletion/ retention labels of the node words, denoted as $D = \{(T_j, S_j)\}_{j=1}^{N}$.

## 3 Approach

This section discusses our tree-based neural models for node label prediction and subsequent sentence compression.

### 3.1 Base Model : Bi-LSTM

Inspired by Wang et al. [23], our base model takes an input sentence as a sequence of words $w_1, w_2, w_3, ...., w_n$. The model utilizes the sequential context representation of a word computed using a Bi-LSTM [9] to decide whether each word in the input sentence is to be retained.

$$x_i = \mathbf{E}_w(w_i) \tag{2}$$

$$\overrightarrow{h_i} = \mathbf{LSTM}_{\overrightarrow{\theta}}(\overrightarrow{h_{i-1}}, x_i) \tag{3}$$

$$\overleftarrow{h_i} = \mathbf{LSTM}_{\overleftarrow{\theta}}(\overleftarrow{h_{i-1}}, x_i) \tag{4}$$

$$h_i = \overrightarrow{h_i} \oplus \overleftarrow{h_i} \tag{5}$$

**LSTM** is computed using update equations

.

$$
\begin{aligned}
&\mathrm{i}_t = \tanh(W_{xi}x_t + W_{hi}h_{t-1} + b_i)\\
&j_t = \sigma(W_{xj}x_t + W_{hj}h_{t-1} + b_j)\\
&f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)\\
&o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)\\
&c_t = f_t \otimes c_{t-1} + i_t \otimes j_t\\
&h_t = \tanh(c_t) \otimes o_t
\end{aligned}
$$

$\theta$ are the parameters of **LSTM** [10] and $\mathrm{E}_w$ is the word embedding lookup table. Using the computed representation $h_j$ of word $w_j$, the label $y_j$ is predicted as

$$P(y_j|h_j) = \mathrm{softmax}(\mathbf{W}h_j + \mathbf{b}) \tag{6}$$

### 3.2 Our TreeLSTM Model

We use the concept of bottom up Tree-LSTM proposed by Tai et al. [19] to compute the dense representation for an input syntactic tree node. Given a tree, let *C(j)* denote the set of children of the node $j$. The Child-Sum Tree-LSTM transitions are:

$$\widetilde{h_j} = \sum_{k \epsilon C(j)} h_k \tag{7}$$

$$i_j = \sigma(\mathbf{W}^{(i)}x_j + \mathbf{U}^{(i)}\widetilde{h_j} + \mathbf{b}^{(i)}) \tag{8}$$

$$f_{jk} = \sigma(\mathbf{W}^{(f)}x_j + \mathbf{U}^{(f)}h_k + \mathbf{b}^{(f)}), k\epsilon C(j) \tag{9}$$

$$o_j = \sigma(\mathbf{W}^{(o)}x_j + \mathbf{U}^{(o)}\widetilde{h_j} + \mathbf{b}^{(o)}) \tag{10}$$

$$u_j = \tanh(\mathbf{W}^{(u)}x_j + \mathbf{U}^{(u)}\widetilde{h_j} + \mathbf{b}^{(u)}) \tag{11}$$

$$c_j = i_j \odot u_j + \sum_{k \epsilon C(j)} f_{jk} \odot c_k \tag{12}$$

$$h_j = o_j \odot \tanh(c_j) \tag{13}$$

where $x_j$ is the input at each node step, $\sigma$ denotes the logistic sigmoid function and $\odot$ denotes elementwise multiplication. Input $x_j\ \epsilon\mathrm{R}^d$ is the embedding of word $w_j$ at the node $j$.
Using the dense representation $h_j$ of node $j$, the label $y_j$ is predicted as

$$P(y_j|h_j) = \mathrm{softmax}(\mathbf{W}h_j + \mathbf{b}) \tag{14}$$

$W^{(i)}$, $U^{(i)}$, $b^{(i)}$, $W^{(f)}$, $U^{(f)}$, $b^{(f)}$, $W^{(o)}$, $U^{(o)}$, $b^{(o)}$, $W^{(u)}$, $U^{(u)}$, $b^{(u)}$, $W$ and $b$ are model parameters which are optimized using training dataset.

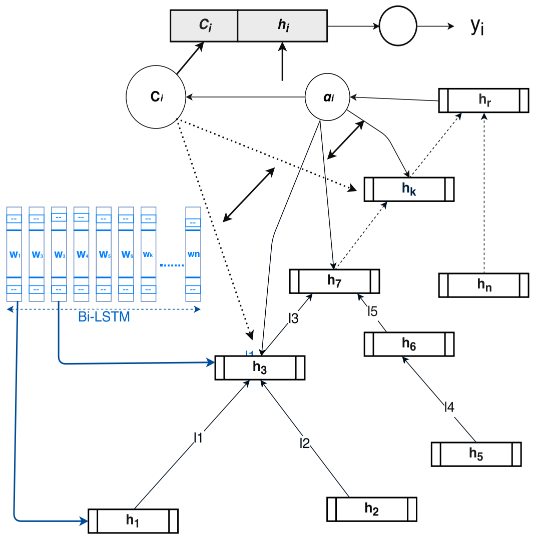

**Fig. 3.** Tree LSTM states are represented on the right side. Double ended arrows represents self attention based context vector computation.

## 3.3 Tree LSTM Model + Bi-LSTM

We further combine sequential context and tree context to compute the dense representation of a node. This is achieved by taking sequential context vector of the word as node input in TreeLSTM. Let $w_1$, $w_2$,..., $w_n$ be the sequence of words in the sentence from which the input syntactic tree is parsed out. Also $t_1$, $t_2$,..., $t_n$ be the corresponding POS tag sequence of words. We compute the dense representation of a word in the sequential context using Bi-LSTM [9] as

$$xs_i = \mathbf{E}_w(w_i) \oplus \mathbf{E}_t(t_i) \tag{15}$$

$$\overrightarrow{hs_i} = \mathbf{LSTM}_{\overrightarrow{\theta}}(\overrightarrow{hs_{i-1}}, xs_i) \tag{16}$$

$$\overleftarrow{hs_i} = \mathbf{LSTM}_{\overleftarrow{\theta}}(\overleftarrow{hs_{i-1}}, xs_i) \tag{17}$$

$$hs_i = \overrightarrow{hs_i} \oplus \overleftarrow{hs_i} \tag{18}$$

$\mathrm{E}_w(w_i)$ $\epsilon \mathrm{R}^d$ and $\mathrm{E}_t(t_i)$ $\epsilon \mathrm{R}^t$ are word and tag embeddings, respectively, $\oplus$ represents concatenation of vectors and $\overleftarrow{\theta}$ and $\overrightarrow{\theta}$ are parameters of the Bi-LSTM model. The computed sequential context representation $hs_j$ computed using Equation 18 for the word $w_j$ is used as the input $x_j$ (Equations 8, 9, 10 and 11) at the corresponding node $j$ in TreeLSTM. Using the computed node representation, label $y_j$ is predicted using Equation 14.

### 3.4 Tree LSTM Model + Label Features

Tracking dependency labels in TreeLSTM is crucial for effectively defining the syntactic context of a node. Along with sequential context, the current model incorporates dependency labels to the parent in the node input. The node input in TreeLSTM (equations 8, 9, 10 and 11) is computed as

$$x_j = \mathbf{E}_{dl}(dl_j) \oplus hs_j \tag{19}$$

where $\mathbf{E}_{dl}(dl_j)\ \epsilon \mathrm{R}^{dl}$ is the embedding for dependency label of node $j$ to the parent node and $hs_j$ sequential context representation of word $w_j$ computed using Equation 18. After computing each node representation, the label of the node is predicted using Equation 14.

### 3.5 Self Attention Over Context Nodes for Computing Context Vector

The representation of syntax context of each node is computed by attending over its context nodes. Formally, context nodes of node $i$ are defined as follows

$\mathit{TopContextNodes}_{i,dt}$ ←All nodes within a path distance $dt$ towards root from node $i$

$\mathit{BottomContextNodes}_{i,db}$ ←All nodes within a path distance $db$ towards leaves from node $i$

In Figure 2, for the node '*People*', the top context nodes are along the blue line and bottom context nodes are along the red line. $dt$ and $db$ are constants, which are optimized emprically. $\mathit{ContextNodes}_i$ is the union of $\mathit{TopContextNodes}_{i,dt}$ of $\mathit{BottomContextNodes}_{i,db}$.

The overall network architecture is shown in Figure 3. Node representations are computed using the same method as described in section 3.4. The context vector $C_i$ for node $i$ is computed by weighting the context nodes with the attention weights computed based on the representation of node $i$.

$$C_i \leftarrow \sum_j \beta_j h_j, j \in ContextNodes_j \tag{20}$$

$$\beta_j \leftarrow \mathrm{softmax}(\alpha_j) \tag{21}$$

$$\alpha_j \leftarrow attend(h_j, h_i) \tag{22}$$

where $W_a$ is a model parameter, $h_i$ is the dense representation for node $i$, $\beta_j$ is the attention weight for context node $j$. The label of the node $i$ is predicted as follows,

$$P(y_i|h_i) \leftarrow \mathrm{softmax}(\mathbf{W}(C_i \oplus h_i) + \mathbf{b}) \tag{23}$$

## 4 Training

The training goal is to minimize the average negative-log likelihood loss in predicting the label of each node for each tree.

$$\text{Minimize:} \frac{-\sum_i log(P(y_{i_a}))}{N}, \tag{24}$$

where $y_{i_a}$ is the actual label of node $i$ and $N$ is the number of nodes in the syntactic tree.

## 5 Maximum Density Subtree Cut Algorithm for Sentence Simplification

For a tree $T$ with weights set for all its nodes, density is defined as the average weight of nodes. Maximum density subtree cut algorithm extracts the subtree with maximum density value within given size limits. We define the density of syntactic dependency tree $T$ as follows,

$$Density(T) \leftarrow \frac{\sum_{j\epsilon Nodes(T)} P(y_i = 1)}{N} \tag{25}$$

where *P($y_i$=1)* is the probability for the word at node $i$ to be retained, computed by the neural network as represented in Equation 23, $Nodes(T)$ is the set of nodes in $T$ and $N$ is the total number of nodes in $T$. Consequently, for a given maximum number of nodes, maximum density subtree $T^{'}$ of a syntactic tree contains a set of nodes which are highly probable to be retained. For a syntactic dependency tree $T$ with $N$ nodes, the pruning process can be represented as follows.

$$T^{'} \leftarrow getMaxDenSubtree(T, c * N, q * N)(26)$$

where $c$ and $q$ are constants which are empirically optimized. We use a greedy

maximum density subtree cut algorithm ensures that a node with a word is essential to maintain grammaticality as decided by its relation with parent node is not removed without removing its parent, irrespective of their *P($y_j$=1)* value. We list the dependency relations *nsubj, csubj, nsubjpass, xsubj, aux, xcomp, pobj, acomp, dobj, case, det, poss, possessive, auxpass ,ccomp, neg, expl, cop, prt, mwe, pcomp, iobj, number, quantmod, predet, dep*, and *mark* as essential to maintain grammaticality. During each iteration, the algorithm searches for the next subtree to be pruned out within grammatical constraints while enforcing the size constraints.

## 6 Experiments

In order to validate the cross-domain effectiveness of our model, we used two different different datasets representing two different domains for training and evaluation.

| *dt* | *db* | **F1- Measure** |
|---|---|---|
| 1 | 3 | 0.81 |
| 3 | 4 | 0.81 |
| 4 | 3 | **0.83** |
| 4 | 5 | 0.82 |
| 6 | 3 | 0.81 |
| 6 | 6 | 0.80 |
| 3 | 4 | 0.82 |

**Table 1.** Tuning *dt* and *db* dataset

- **Google News Data Set :** The parallel dataset[4] released by Filippova et al. [6] contains 10000 sentences and corresponding compressed version.
- **BNC News :** The second dataset consists of 1500 sentences taken from British National Corpus (BNC) and the American News Text corpus before 2008 and their ground truth compressed versions. The dataset[5] is collected and released by Cohn et al. [4].

We split the Google News dataset into training (1001-9000), testing (1-1000) and validation sets (9000-10000). The offset and size of each set remain exactly same as those of Wang et al., [23]. The BNC News dataset is utilized as a cross-domain testset.

## 6.1 Experimental Settings

We evaluate our approaches using F1 score and accuracy. The former is derived from precision and recall values, where precision is defined as the percentage of retained words that overlap with the ground truth and recall is defined as the percentage of words in the ground truth compressed sentences that overlap with the generated compressed sentences. The latter is defined as the percentage of tokens for which the predicted label $y_i$ is correct.

We evaluate five different variations of our method.

- **TreeLSTM :** A bottom up TreeLSTM [19] taking word embedding as input at each node as described in Section 3.2.
- **TreeLSTM + Bi-LSTM:** A bottom up TreeLSTM taking sequential context representation of a word computed using a Bi-LSTM as input at each node as described in Section 3.3.
- **TreeLSTM + Bi-LSTM + Dependency Features:** A bottom up TreeLSTM taking the concatenation of sequential context representation of a word and dependency label embedding as input at each node as described in Section 3.4.
- **TreeLSTM + Bi-LSTM + Self Attention:** In this setting, to estimate the probability for each node to be retained, we take the weighted context

[4] http://storage.googleapis.com/sentencecomp/compression-data.json
[5] http://jamesclarke.net/research/resources/

| **Method** | **Size** | **GN** | | | **BNC** | | |
|---|---|---|---|---|---|---|---|
| | | F1 | Acc | CR | F1 | Acc | CR |
| LSTM[6] | 2M | 0.80 | - | 0.39 | - | - | - |
| LSTM+[6] | 2M | **0.82** | - | 0.38 | - | - | - |
| Abstractive seq2seq (ala [23]) | 3.8M | 0.09 | 0.02 | 0.16 | 0.14 | 0.06 | 0.21 |
| LSTM (baseline [23]) | 8K | 0.74 | 0.75 | 0.45 | 0.51 | 0.48 | 0.37 |
| LSTM+ (baseline [23]) | 8K | 0.77 | 0.78 | 0.47 | 0.54 | 0.51 | 0.38 |
| BiLSTM [23] | 8K | 0.75 | 0.76 | 0.43 | 0.52 | 0.50 | 0.34 |
| BiLSTM+SynFeat [23] | 8K | 0.80 | 0.82 | 0.43 | 0.57 | 0.54 | 0.37 |
| **Our Models** | | | | | | | |
| TreeLSTM | 8K | 0.73 | 0.72 | 0.46 | 0.49 | 0.47 | 0.33 |
| TreeLSTM + Bi-LSTM | 8K | 0.80 | 0.81 | 0.44 | 0.53 | 0.53 | 0.36 |
| TreeLSTM + Bi-LSTM + Dep | 8K | 0.80 | 0.835 | 0.42 | 0.57 | 0.54 | 0.36 |
| TreeLSTM + Bi-LSTM + Attention | 8K | 0.81 | **0.845** | 0.43 | 0.59 | 0.54 | 0.36 |
| **Syntactic Constraints** | | | | | | | |
| Traditional ILP[3] | N/A | 0.54 | 0.56 | 0.62 | 0.64 | 0.56 | 0.56 |
| BiLSTM+SynFeat+ILP [23] | 8K | 0.78 | 0.78 | 0.57 | 0.66 | 0.58 | 0.53 |
| TreeLSTM+Attn+ MDT(CR= 0.21) | 8K | 0.67 | 0.66 | 0.21 | 0.42 | 0.41 | 0.21 |
| TreeLSTM+Attn+ MDT(CR=0.38) | 8K | 0.81 | 0.83 | 0.38 | 0.59 | 0.53 | 0.38 |
| TreeLSTM+Attention+ MDT | 8K | 0.79 | 0.79 | 0.55 | **0.70** | **0.60** | 0.54 |

**Table 2.** Comparison with base systems: F1, Acc - Accuracy, CR - Compression Ratio, GN - Google News

| | **RD** | **IF** |
|---|---|---|
| Traditional ILP | 3.3 | 3.27 |
| BiLSTM+SynFeat+ILP | 4.21 | 4.1 |
| TreeLSTM + Self Attention+ MDT | **4.30** | **4.25** |

**Table 3.** Human Evaluation

of the node into account as described in Section 3.5. Remaining settings are same as the setting described above.

– **TreeLSTM + Self Attention+ MDT:** In this method, $P(y_i = 1)$ for each node $i$ is computed using the method explained above. Subsequently, maximum density subtree (MDT) cut algorithm is applied as explained in Section 5 to decide the final label of each node.

**Settings:** Our model was trained using Adam [13] with the learning rate initialized at 0.001. The TreeLSTM hidden layer dimension is set to 200. The dimension of the hidden layers of bi-LSTM is 100. Word embeddings are initialized from GloVe 100-dimensional pre-trained embeddings [16]. POS and dependency embeddings are randomly initialized with 40-dimensional vectors. Word embeddings are set as updatable during training. The batch size is set as 20. Constants $c$, $q$ (equation 26), $dt$, and $db$ (section 3.5) are set to 0.7, 0.3, 4 and 3 respectively for maximum accuracy in validation data using grid-search. The Table 1 list the accuracy for different values of *dt* and *db* in development dataset. We used Satndford Parser[6] for creating syntactic parse trees.

## 6.2 Results

Table 2 shows the performance of our approaches, the method of [23] and other baselines. The models include BiLSTM without incorporating any syntactic feature, BiLSTM+SynFeat in which they incorporate syntactic features in a BiLSTM and BiLSTM+SynFeat+ILP in which they use ILP to predict the final label $y$ for each word in the input sentence. Their baselines are LSTM [6], LSTM+ in which syntactic features are incorporated with LSTM, Traditional ILP [3] and Abstractive seq2seq which is an abstractive sequence-to-sequence [7] model trained on 3.8 million Gigaword title-article pairs [15]

**Effectiveness of Neural Tree Model** A simple bottom-up TreeLSTM with word-embedding as input does not yield good results. We observe that explicit use of syntactic features is necessary for defining the syntactic context of a node. The performance dramatically increased when sequential context representation consisting of word and POS tag information is used as input to the TreeLSTM at each node (TreeLSTM + Bi-LSTM). TreeLSTM, which jointly tracks sequential context representation of a word and dependency label while computing node representations (TreeLSTM + Bi-LSTM + Dependency Features), outperforms BiLSTM+SynFeat in terms of accuracy in Google News dataset and yields comparable results in BNC dataset. This shows that, a TreeLSTM with explicit use of syntactic features can leverage the syntactic context better than Bi-LSTM utilizing syntactic features (BiLSTM+SynFeat).

Computing context vector representation (TreeLSTM + Bi-LSTM + Self Attention) and subsequent labelling of nodes outperforms BiLSTM+SynFeat in all domains Table 2. Our observation is that computing context vector representation using self attention can effectively model the syntactic context of a node and identifies the regions of a syntactic tree, which holds abstract information relevant for sentence compression.

**Effectiveness Maximum Density Tree Cut** Incorporation of multi-density subtree extraction with neural model (TreeLSTM + Self Attention+ MDT) outperforms the state-of-art BiLSTM+SynFeat+ILP on both of Google News and BNC datasets, proving its efficiency in cross-domain application. This is probably because it locates regions of a syntactic dependency tree holding retainable abstract information, explicitly relying on dependency relations to maintain grammaticality. As a result, the approach enjoys more flexibility in the search for abstract content than an ILP method with hard constraints. Also, self attention based context vector computation can identify patterns of informativeness more accurately.

[6] https://nlp.stanford.edu/software/lex-parser.shtml
[7] http://opennmt.net/

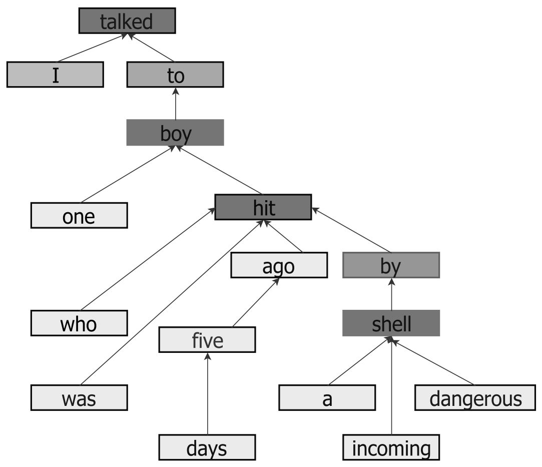

**Fig. 4.** Tree Node Weights: Grey scale shows the probability to be retained

**Evaluating Cross Domain Effectiveness by Fixing Compression Ratio** The sentence compressions in BNC dataset is relatively larger than Gigaword title-article pairs [15]. Taking this into account, in order to have a fair comparison with Abstractive seq2seq, we have tested our MDT model also by pre-fixing the model with their compression ratio in the BNC dataset as shown in Table 2. We have also tested the MDT model by setting the compression ration of LSTM+ in the BNC datset. In both cases, MDT has a better score than the two baselines systems, which rules out the chance for higher compression ratio being the reason for better results in new domains. Approaches with syntactic constraints can ensure a fare comparison as all of them share a similar compression ratio.

## 7 Human Evaluation

90 randomly chosen source sentences and corresponding compressions produced by TreeLSTM + Self Attention+ MDT, BiLSTM+SynFeat+ILP and Traditional ILP are chosen for human evaluation. These three approaches exhibits almost similar average compression ratio and don't enjoy any advantage due to higher compression ratio ensuring a level comparison. The human raters were asked to rate the informativeness (**IF**) and readability (**RD**) of the compressed outputs on a scale of 5. The compressed sentences take a random order during each rating to avoid any kind of bias. The average scores obtained are shown in the Table 3. TreeLSTM + Self Attention+ MDT yields scores which are comparable with those of BiLSTM+SynFeat+ILP. The results show that MDT with constraints on grammaticality can better preserve readability than Traditional ILP. None of the methods has an advantage due to compression ratio as all of them exhibit similar average compression compression ratio in test

## 8 Related Work

Our work belongs to the line of extractive sentence compression approaches. A seminal graph-based sentence compression method was suggested by Mcdonald et al. [14]. They assign a score for each word pair existing in the original sentence and search for a compressed sentence with the maximum total score within a given length limit. Clarke et al. [3] uses an Integer Linear Programming (ILP) framework for sentence compression. They compute a relevance score for each word and then incorporates the scores in the ILP formulation for ranking candidates.

Filippova et al. [7] apply ILP over syntactic dependency trees to trace-out a proper subtree corresponding to a grammatically correct simplified sentence. Berg et al. [1] use a joint model to extract and compress for multi-document summarization. Their approach weighs bi-grams using a supervised linear model. In contrast, our method uses neural network to estimate informativeness, while utilizing generalizability of syntactic tree based approaches.

There has been work in the past which tried to impose syntactic constraints via soft logic [12, 25] or by hard structural rules [22, 21, 17, 24]. However, a combination of neural methods and multi-density tree cut algorithm can more flexibly search for compressed representation of the source sentence. Also, the current work investigates domain adaptability of sentence simplification methods by an optimum combination of neural methods and syntactic tree based approaches.

There has been work which builds neural network models trained on large datasets, both for extractive [6] and abstractive [2, 18] sentence compression. However, these techniques require a large amount of training data. They also tend to overfit in the domain of training data and end up with a poor performance in a cross-domain settings [23]. There are has been previous works to improve text simplification techniques such as multi- document summarization[11] and headline generation[20]. However scope and challenges of extractive sentence summarization is different from these problems.

The most recent work, which effectively merge the potential strengths of syntax based approaches and data-driven neural network model, is Wang et al. [23]. They make significant improvement in cross-domain sentence compression with a relatively smaller training set. However, they have predominantly relied on the sequential context of a word along with syntactic information to decide whether a word needs to be retained or not, despite that the syntactic context of word is better definable within a syntactic tree. Our approach decides whether to retain or to remove a word based on its context within a dependency tree. In this sense, we extend the work of Wang et al. [23] by incorporating the strength of traditional syntax tree noise-pruning methods by using a maximum density subtree extraction algorithm.

## 9 Conclusion

We investigated an approach for sentence compression which utilize the possibilities of neural models in syntactic tree pruning for sentence compression. Our method yields the best results in terms of F1-measure and accuracy in two different domains proving its domain adaptability. There is scope for research in future for a method which jointly learn to compute weights and extract subtrees. Parsing errors can be overcome by using the top-K parse trees generated by the parser and train the current approach using all the trees.

## References

1. Berg-Kirkpatrick, T., Gillick, D., Klein, D.: Jointly learning to extract and compress. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. pp. 481–490. Association for Computational Linguistics (2011)
2. Chopra, S., Auli, M., Rush, A.M.: Abstractive sentence summarization with attentive recurrent neural networks. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 93–98 (2016)
3. Clarke, J., Lapata, M.: Global inference for sentence compression: An integer linear programming approach. Journal of Artificial Intelligence Research **31**, 399–429 (2008)
4. Cohn, T., Lapata, M.: Large margin synchronous generation and its application to sentence compression. In: EMNLP-CoNLL. pp. 73–82 (2007)
5. Cohn, T.A., Lapata, M.: Sentence compression as tree transduction. Journal of Artificial Intelligence Research **34**, 637–674 (2009)
6. Filippova, K., Alfonseca, E., Colmenares, C.A., Kaiser, L., Vinyals, O.: Sentence compression by deletion with lstms. In: EMNLP. pp. 360–368 (2015)
7. Filippova, K., Strube, M.: Dependency tree based sentence compression. In: Proceedings of the Fifth International Natural Language Generation Conference. pp. 25–32. Association for Computational Linguistics (2008)
8. Galanis, D., Androutsopoulos, I.: An extractive supervised two-stage method for sentence compression. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. pp. 885–893. Association for Computational Linguistics (2010)
9. Graves, A., Jaitly, N., Mohamed, A.r.: Hybrid speech recognition with deep bidirectional lstm. In: Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on. pp. 273–278. IEEE (2013)
10. Hochreiter, S., Schmidhuber, J.: Lstm can solve hard long time lag problems. In: Advances in neural information processing systems. pp. 473–479 (1997)
11. Hong, K., Conroy, J.M., Favre, B., Kulesza, A., Lin, H., Nenkova, A.: A repository of state of the art and competitive baseline summaries for generic news summarization. In: LREC. pp. 1608–1616 (2014)
12. Huang, M., Shi, X., Jin, F., Zhu, X.: Using first-order logic to compress sentences. In: Aaai (2012)
13. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
14. McDonald, R.T.: Discriminative sentence compression with soft syntactic evidence. In: Eacl (2006)
15. Napoles, C., Gormley, M., Van Durme, B.: Annotated gigaword. In: Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction. pp. 95–100. Association for Computational Linguistics (2012)

16. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
17. Qian, X., Liu, Y.: Polynomial time joint structural inference for sentence compression. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). vol. 2, pp. 327–332 (2014)
18. Rush, A.M., Chopra, S., Weston, J.: A neural attention model for abstractive sentence summarization. arXiv preprint arXiv:1509.00685 (2015)
19. Tai, K.S., Socher, R., Manning, C.D.: Improved semantic representations from tree-structured long short-term memory networks (2015)
20. Takase, S., Suzuki, J., Okazaki, N., Hirao, T., Nagata, M.: Neural headline generation on abstract meaning representation. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. pp. 1054–1059 (2016)
21. Thadani, K.: Approximation strategies for multi-structure sentence compression. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 1241–1251 (2014)
22. Thadani, K., McKeown, K.: Sentence compression with joint structural inference. In: Proceedings of the Seventeenth Conference on Computational Natural Language Learning. pp. 65–74 (2013)
23. Wang, L., Jiang, J., Chieu, H.L., Ong, C.H., Song, D., Liao, L.: Can syntax help? improving an lstm-based sentence compression model for new domains. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 1385–1393 (2017)
24. Yao, J.g., Wan, X.: Greedy flipping for constrained word deletion. In: AAAI. pp. 3518–3524 (2017)
25. Yoshikawa, K., Hirao, T., Iida, R., Okumura, M.: Sentence compression with semantic role constraints. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2. pp. 349–353. Association for Computational Linguistics (2012)