The first technique employs ‘unbiased’ data; i.e., randomly shuffled sentences of the source document, to pretrain the model. The second technique uses an auxiliary ROUGE-based loss that encourages the model to distribute importance scores throughout a document by mimicking sentence-level ROUGE scores on the training data. We show that these techniques significantly improve the performance of a competitive reinforcement learning based extractive system, with the auxiliary loss being more powerful than pretraining.

1 Introduction

Extractive summarization remains a simple and fast approach to produce summaries which are grammatical and accurately represent the source text. In the news domain, these systems are able to use a dominant signal: the position of a sentence in the source document. Due to journalistic conventions which place important information early in the articles, the lead sentences often contain key information. In this paper, we explore how systems can look beyond this simple trend.

Naturally, automatic systems have all along exploited position cues in news as key indicators of important content (Schiffman et al., 2002; Hong and Nenkova, 2014; Liu, 2019). The ‘lead’ base-
the training articles. We use this shuffled dataset for pre-training, followed by training on the original (unshuffled) articles. The second method introduces an auxiliary loss which encourages the model’s scores for sentences to mimic an estimated score distribution over the sentences, the latter computed using ROUGE overlap with the gold standard. We implement these techniques for two recent reinforcement learning based systems, RNES (Wu and Hu, 2018) and BanditSum (Dong et al., 2018), and evaluate them on the CNN/Daily Mail dataset (Hermann et al., 2015).

We find that our auxiliary loss achieves significantly better ROUGE scores compared to the base systems, and that the improvement is even more pronounced when the true best sentences appear later in the article. On the other hand, the pre-training approach produces mixed results. We also confirm that when summary-worthy sentences appear late, there is a large performance discrepancy between the oracle summary and state-of-the-art summarizers, indicating that learning to balance lead bias with other features of news text is a noteworthy issue to tackle.

2 Related Work

Modern summarization methods for news are typically based on neural network-based sequence-to-sequence learning (Kalehbrunner et al., 2014; Kim, 2014; Chung et al., 2014; Yin and Pei, 2015; Cao et al., 2015; Cheng and Lapata, 2016; Nallapati et al., 2017; Narayan et al., 2018a; Zhou et al., 2018). In MLE-based training, extractive summarizers are trained with gradient ascent to maximize the likelihood of heuristically-generated ground-truth binary labels (Nallapati et al., 2017). Many MLE-based models do not perform as well as their reinforcement learning-based (RL) competitors that directly optimize ROUGE (Paulus et al., 2018; Narayan et al., 2018b; Dong et al., 2018; Wu and Hu, 2018). As RL-based models represent the state of the art for extractive summarization, we analyze them in this paper.

The closest work to ours is a recent study by Kedzie et al. (2018) which showed that MLE-based models learn a significant bias for selecting early sentences when trained on news articles as opposed to other domains. As much as 58% of selected summary sentences come directly from the lead. Moreover, when these models are trained on articles whose sentences are randomly shuffled, the performance drops considerably for news domain only. While this drop could be due to the destruction of position cues, it may also arise because the article’s coherence and context were lost.

In this paper, we employ finer control on the distortion of sentence position, coherence, and context, and confirm that performance drops are mainly due to the lack of position cues. We also propose the first techniques to counter the effects of lead bias in neural extractive systems.

3 Base Models for Extractive Summarization

In supervised systems, given a document $D = \{s_1, \ldots, s_n\}$ with $n$ sentences, a summary can be seen as set of binary labels $y_1, \ldots, y_n \in \{0, 1\}$, where $y_i = 1$ indicates that the $i$-th sentence is included in the summary.

We choose to experiment with two state-of-the-art RL-based extractive models: RNES (Wu and Hu, 2018) and BanditSum (Dong et al., 2018). Both employ an encoder-decoder structure, where the encoder extracts sentence features into fixed-dimensional vector representations $h_1, \ldots, h_n$, and a decoder produces the labels $y_1, \ldots, y_n$ based on these sentence representations. RNES uses a CNN+bi-GRU encoder, and BanditSum a hierarchical bi-LSTM. RNES’s decoder is auto-regressive, meaning it predicts the current sentence’s label based on decisions made on previous sentences; i.e., $y_t = f(D, h_t, y_1:t-1)$. In BanditSum, there is no such dependence: it produces affinity scores for each sentence and the top scoring sentences are then selected.

4 Lead Bias of News Systems

First, we investigate the impact of sentence position on our models. We manipulate the original CNN/Daily Mail dataset to preserve sentence position information at different levels. In the random setting, sentences are shuffled randomly; in reverse, they are in reverse order; in insert-lead and insert-lead3, we insert an out-of-document sentence (chosen randomly from the corpus) as the first sentence or randomly as one of the first three sentences, respectively.

In Table 2, we show BanditSum’s performance,\(^1\) when trained and tested on the various datasets. All models (except random) perform

\(^1\)We notice the same trends on RNES.
Table 2: BanditSum’s performance—calculated as the average between ROUGE-1, -2, and -L F1—on the validation set of the CNN/Daily Mail corpus. The sentence position information is perturbed at different levels, as explained in Section 4.

<table>
<thead>
<tr>
<th></th>
<th>original</th>
<th>random</th>
<th>reverse</th>
<th>insert-lead</th>
<th>insert-lead3</th>
<th>Mean</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lead-3 baseline</td>
<td>32.68</td>
<td>22.81</td>
<td>17.94</td>
<td>27.67</td>
<td>27.68</td>
<td>25.76</td>
<td>5.00</td>
</tr>
<tr>
<td>original</td>
<td>33.85</td>
<td>26.18</td>
<td>20.70</td>
<td>28.71</td>
<td>31.59</td>
<td>29.09</td>
<td>5.12</td>
</tr>
<tr>
<td>random</td>
<td>30.88</td>
<td>29.70</td>
<td>21.63</td>
<td>29.97</td>
<td>31.59</td>
<td>29.09</td>
<td>4.93</td>
</tr>
<tr>
<td>reverse</td>
<td>21.35</td>
<td>26.32</td>
<td>33.59</td>
<td>28.81</td>
<td>31.59</td>
<td>24.91</td>
<td>4.57</td>
</tr>
<tr>
<td>insert-lead</td>
<td>33.21</td>
<td>26.07</td>
<td>20.70</td>
<td>33.41</td>
<td>31.59</td>
<td>29.00</td>
<td>4.93</td>
</tr>
<tr>
<td>insert-lead3</td>
<td>32.29</td>
<td>25.57</td>
<td>20.22</td>
<td>32.15</td>
<td>28.63</td>
<td>28.63</td>
<td>4.98</td>
</tr>
</tbody>
</table>

worse when tested on a mismatched data perturbation. Even when the distortion is at a single lead position in insert-lead and insert-lead3, the performance on the original data is significantly lower than when trained without the distortion. These results corroborate Kedzie et al. (2018)’s findings for RL-based systems. Interestingly, the random model has the best mean performance and the lowest variation indicating that completely removing the position bias may allow a model to focus on learning robust sentence semantics.

5 Learning to Counter Position Bias

We present two methods which encourage models to locate key phrases at diverse parts of the article.

5.1 Multi-Stage Training

This technique is inspired by the robust results from the random model in section 4. We implement a multi-stage training method for both BanditSum and RNES where in the first few epochs, we train on an ‘unbiased’ dataset where the sentences in every training document are randomly shuffled. We then fine-tune the models by training on the original training articles. The goal is to prime the model to learn sentence semantics independently of position, and then introduce the task of balancing semantics and positional cues.

5.2 ROUGE-based Auxiliary Loss

We observed that BanditSum tends to converge to a low-entropy policy, in the sense that the model’s affinity scores are either 1 or 0 at the end of training. Furthermore, over 68% of its selections are from the three leading sentences of the source. Regularizing low-entropy policies can increase a model’s propensity to explore potentially good states or stay close to a known good policy (Nachum et al., 2017; Galashov et al., 2019). We extend this idea to summarization by introducing a ROUGE-based loss which regularizes the model policy using an estimate of the value of individual sentences.

These sentence-level estimates are computed as a distribution $P_R$:

$$P_R(x = i) = \frac{r(s_i, G)}{\sum_{j=1}^{n} r(s_j, G)}$$

where $r$ is the average of ROUGE-1, -2 and -L F1 scores between sentence $s_i$ in the article and the reference summary $G$. We would like the model’s predictive distribution $P_M$ to approximately match $P_R$. To compute $P_M$, we normalize the predicted scores from a non-auto-regressive model. In an auto-regressive model such as RNES, the decision of including a sentence depends on those selected so far. So a straightforward KL objective is hard to implement, and we use this technique for BanditSum only.

Our auxiliary loss is defined as the KL divergence: $L_{KL} = D_{KL}(P_R \parallel P_M)$. The update rule then becomes:

$$\theta^{t+1} = \theta^{(t)} + \alpha \left( \nabla L_M(\theta^{(t)}) + \beta \nabla L_{KL}(\theta^{(t)}) \right)$$

where $\theta^{(t)}$ represents the model’s parameters at time step $t$, $L_M$ is the original model’s loss function, and $\beta$ is a hyperparameter.

6 Experimental Setup

We use the CNN/Daily Mail dataset (Hermann et al., 2015) with the standard train/dev/test splits of 287,227/13,368/11,490. To avoid inconsistencies, we built on top of the author-provided implementations for BanditSum and our faithful reimplementation of RNES.

To reduce training time, we pre-compute and store the average of ROUGE-1, -2, and -L for every sentence triplet of each article, using a HDF5 table and PyTables (PyTables Developers Team, 2002-2019; The HDF Group, 1997-2019). This allows for a considerable increase in training speed.
Table 3: ROUGE scores for systems. ‘Overlap’ denotes the model’s overlap in extraction choices with the lead-3 baseline. Scores significantly higher than BanditSum with \( p < 0.001 \) (bootstrap resampling test) are marked with *.

We limit the maximum number of sentences considered in an article to the first 100.

All the models were trained for 4 epochs. For the multi-stage training, we pretrain for 2 epochs, then train on the original articles for 2 epochs. We set the auxiliary loss hyperparameters \( \alpha = 1e^{-4} \) and \( \beta = 0.0095 \) in eq. 2 based on a grid search using the Tune library (Liaw et al., 2018).

We also train a baseline entropy model by replacing \( \mathcal{L}_{KL} \) with the negated entropy of \( P_M \) in eq. 2. This loss penalizes low entropy, helping the model explore, but it is ‘undirected’ compared to our proposed method. We present the results of Lead-3 baseline (first 3 sentences), and two other competitive models—Refresh\(^2\) (Narayan et al., 2018a) and NeuSum (Zhou et al., 2018).

Lastly, we include results from an oracle summarizer, computed as the triplet of source sentences with the highest average of ROUGE-1, -2 and -L scores against the abstractive gold standard.

7 Results and Discussion

Table 3 reports the F1 scores for ROUGE-1, -2 and -L (Lin, 2004). We use the \texttt{pyrouge}\(^3\) wrapper library to evaluate the final models, while training with a faster Python-only implementation\(^4\).

We test for significance between the baseline models and our proposed techniques using the bootstrap method. This method was first recommended for testing significance in ROUGE scores by Lin (2004), and has subsequently been advocated as an appropriate measure in works such as Dror et al. (2018) and Berg-Kirkpatrick et al. (2012).

The simple entropy regularizer has a small but not significant improvement, and pretraining has a similar improvement only for RNES. But the auxiliary ROUGE loss significantly (\( p < 0.001 \)) improves over BanditSum, obtaining an extra 0.15 ROUGE points on average. The last column reports the percentage of summary sentences which overlap with the lead. The auxiliary loss leads to a 4.7% absolute decrease in such selections compared to the base system, while also reaching a better ROUGE score. Figure 1 shows that the reward (average ROUGE-1, -2, -L) for the auxiliary loss model is consistently above the base.

We also examined the auxiliary loss model on documents where the summary is mostly comprised of lead sentences \( D_{early} \), mostly sentences much later in the article \( D_{late} \), and a dataset at the midway point, \( D_{med} \). To create these sets, we rank test articles using the average index of its summary sentences in the source document. The 100 test articles with lowest average index are \( D_{early} \), the 100 with highest value are \( D_{late} \) and the 100 closest to the median are \( D_{med} \). In Table 4, we can see that the auxiliary loss model’s improvements are even more amplified on \( D_{med} \) and \( D_{late} \).

On the other hand, our pretraining results are mixed. We hope to employ more controlled multitasking methods (Kiperwasser and Ballesteros, 2018) in the future to deal with the issue.

The second line in Table 4 reports the oracle ROUGE scores of the best possible extractive summary. While all systems are quite close to the oracle on \( D_{early} \) they only reach half the performance on \( D_{late} \). This gap indicates that our improvements only scratch the surface, but also that this problem is worthy and challenging to explore.

It is worth noting that we have attempted to build a single model which can summarize both
lead-biased articles and those whose information is spread throughout. Our aim was to encourage the model to explore useful regions as a way of learning better document semantics. But we hypothesize that our models can be further improved by learning to automatically predict when the lead paragraph suffices as a summary, and when the model should look further in the document.

8 Conclusion

In this paper, we have presented the first approaches for learning a summarization system by countering the strong effect of summary-worthy lead sentences. We demonstrate that recent summarization systems over-exploit the inherent lead bias present in news articles, to the detriment of their summarization capabilities. We explore two techniques aimed at learning to better balance positional cues with semantic ones. While our auxiliary loss method achieves significant improvement, we note that there is a large gap which better methods can hope to bridge in the future.

One approach, building on ours, is to examine other ways to combine loss signals (Finn et al., 2017), and to encourage exploration (Haarnoja et al., 2018). We will also carry out deeper study of the properties of \( D_{\text{early}} \) and \( D_{\text{late}} \) type documents and use them to inform new solutions. On cursory analysis, the most frequent terms in \( D_{\text{early}} \) tend to be about UK politics, while in \( D_{\text{late}} \) they are often related to British soccer.

Acknowledgments

This work is supported by the Natural Sciences and Engineering Research Council of Canada, the Institute for Data Valorisation (IVADO), Compute Canada, and the CIFAR Canada AI Chair program.

References


<table>
<thead>
<tr>
<th>Model</th>
<th>( D_{\text{early}} )</th>
<th>( D_{\text{med}} )</th>
<th>( D_{\text{late}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lead-3</td>
<td>46.17</td>
<td>30.90</td>
<td>20.18</td>
</tr>
<tr>
<td>Oracle</td>
<td>50.52</td>
<td>47.92</td>
<td>42.21</td>
</tr>
<tr>
<td>RNES</td>
<td>41.76</td>
<td>32.11</td>
<td>20.62</td>
</tr>
<tr>
<td>RNES+pretrain</td>
<td>41.66</td>
<td>32.38</td>
<td>20.64</td>
</tr>
<tr>
<td>BanditSum</td>
<td>43.10</td>
<td>32.65</td>
<td>21.63</td>
</tr>
<tr>
<td>BanditSum+entropy</td>
<td>41.96</td>
<td>32.59</td>
<td>22.12</td>
</tr>
<tr>
<td>BanditSum+KL</td>
<td>42.63</td>
<td>33.05</td>
<td>21.96</td>
</tr>
</tbody>
</table>

Table 4: Average ROUGE-1, -2 and -L F1 scores on \( D_{\text{early}} \), \( D_{\text{med}} \), \( D_{\text{late}} \). Each set contains 100 documents.


