The Need for Abstractive Summarization

Automatic text summarization is increasingly vital in the digital age. Two approaches:

- **Extractive summarization**: Select and rearrange passages from the original text
- **Abstractive summarization**: Generate novel sentences

Abstractive summarization is essential for high-quality output

Abstractive Summarization with RNNs

- **Recurrent Neural Networks (RNNs)** provide a potentially powerful solution for abstractive summarization.
- Using the attention mechanism (see below), they can generate new words (e.g., “Argentina beat Germany” from the fixed vocabulary “Germany beat Argentina 3-2”) by attending to relevant words (victorious, win).

Easier Copying with Pointer-Generator Network

- For each summary word, the network first calculates the **generation probability** $p_{gen}$.
- $p_{gen}$ interpolates between copying from attention distribution $a$ and generating from vocabulary distribution $P_{vocab}$.

$$P(w) = p_{gen}P_{vocab}(w) + (1 - p_{gen}) \sum_{i \in \text{vocab}} a_i$$

**Advantages:**
- Faster to train
- Easy to accurately reproduce phrases
- Can copy OOV words (don’t need large vocabulary)

Best of both worlds: abstractive (generating) and extractive (copying)

Eliminating Repetition with Coverage

- **Problem**: Summaries are repetitive.
- **Solution**: Penalize repeatedly attending to the same parts of the source text.

On each decoder timestep $t$, the coverage vector $c_t$ tells us what has been attended to (thus summarized) so far.

$$c_t = \sum_{s=0}^{t-1} a_s$$

**Penalize overlap** between coverage vector $c_t$ and new attention distribution $a_t$.

$$\text{covloss}_t = \sum_i \min(a_{t,i}, c_{t,i})$$

Overlap between coverage and current attention

Coverage eliminates undesirable repetition

Experiments

- **Dataset**: CNN/Daily Mail (news article → multi-sentence summary)

<table>
<thead>
<tr>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstractive model (Nallapati et al., 2016)**</td>
<td>35.46</td>
<td>35.01</td>
</tr>
<tr>
<td>pointer-generator &amp; coverage</td>
<td>36.44</td>
<td>35.86</td>
</tr>
<tr>
<td>baseline model</td>
<td>30.32</td>
<td>29.26</td>
</tr>
<tr>
<td>abstractive model (Nallapati et al., 2017)**</td>
<td>30.32</td>
<td>31.70</td>
</tr>
<tr>
<td>pointer-generator &amp; coverage</td>
<td>39.69</td>
<td>36.62</td>
</tr>
</tbody>
</table>

- Our pointer-generator + coverage model beats best abstractive system.
- Extractive systems and lead-3 baseline remain difficult to beat.
- The ROUGE metric is not robust to paraphrasing

Example Output

- **Reference Summary**: 3-0. Germany beat Argentina 2-0 by attending to relevant words (victorious, win).
- **Source Text**: Germany beat Germany beat Germany beat…

How Abstractive Is Our Output?

- Our network uses the **pointer more than the generator** (average $P_{gen} = 0.17$).
- It produces some novel words and phrases, but fewer than the reference summaries.

Open question: How to make pointer-generator network more abstractive?

Conclusion

- Pointer-generator networks enable more accurate copying, are easier to train and can deal with OOVs.
- Coverage drastically reduces repetition.
- ROUGE metric is of limited use for evaluating abstractive systems.
- Future work: make the pointer-generator network more abstractive.