Cutting Down “Fluff”: A Twist on Text Summarization

Kate Salmon (kksalmon1@stanford.edu), Gustavo Torres da Silva (gdasilva@stanford.edu)

Motivation

Context
- People write every day.
- Getting rid of "fluff" (chatter in language) takes time.
- Text summarization is heavily explored, but not as an easy way to assist people in writing more concisely.

Goal
- Given a textual input, output a concise version of that text (i.e. keep the same meaning).
- The 2,225 articles were divided as follows:

<table>
<thead>
<tr>
<th>Split</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percentage</td>
<td>80%</td>
<td>10%</td>
</tr>
</tbody>
</table>

Basic LSTM (Extractive Approach)
- Inputs: embedded text document with its associated binary labels based on the concise version of the text
- Outputs: predictions for whether or not a given word in the fluffed document should appear in the concise version
- Probability that a word is labeled to be in the output text based on model predictions: \( P(w) \approx 0.5 \)
- Based on the binary labels generated, the predicted concise text is generated and compared to the ground truth concise text using ROUGE score.

Loss: Binary Cross Entropy
\[
BCE = -\sum_{i=1}^{N} y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i)
\]

Models

Pointing-Generator (Abstraction Approach)
- Pointing: copies words from source text
- Generator: creates new sentences (seq2seq + attn)
- Flattened version: has probability of generating a word from the vocabulary (as opposed to copying).

\[
P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{i \in Vocab} a_i^2
\]
- Coverage: tracks content covered, avoids repetition.

Loss: finally, the loss combines the coverage loss with the original loss.
\[
loss = -\log P(w) + \lambda \sum_{i \in Vocab} \min(a_i^2, c_i)
\]

Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Score</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Score</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>83.10</td>
<td>22.01</td>
<td>41.66</td>
<td>20.60</td>
<td>78.48</td>
<td>32.76</td>
<td>93.36</td>
<td>90.63</td>
<td>90.99</td>
</tr>
<tr>
<td>PG</td>
<td>80.67</td>
<td>75.49</td>
<td>78.99</td>
<td>55.33</td>
<td>59.34</td>
<td>58.94</td>
<td>94.98</td>
<td>74.92</td>
<td>81.00</td>
</tr>
<tr>
<td>PG + Cov.</td>
<td>94.98</td>
<td>74.92</td>
<td>81.85</td>
<td>71.20</td>
<td>72.53</td>
<td>75.68</td>
<td>94.98</td>
<td>74.92</td>
<td>81.00</td>
</tr>
<tr>
<td>Basic LSTM</td>
<td>93.36</td>
<td>90.63</td>
<td>91.97</td>
<td>84.35</td>
<td>92.37</td>
<td>90.99</td>
<td>93.36</td>
<td>90.63</td>
<td>90.99</td>
</tr>
</tbody>
</table>

Table 1. ROUGE scores on different models.

<table>
<thead>
<tr>
<th>Input</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>the film, showing out of competition in Berlin, is nominated for three oscars, including best actor for Chadwick Boseman, who plays Chedda's wife, Tanita, is nominated for hardy best supporting actress.</td>
<td>the film, showing out of competition in Berlin, is nominated for three oscars, including best actor for Chadwick Boseman, who plays Chedda's wife, Tanita, is nominated for hardy best supporting actress.</td>
</tr>
<tr>
<td>even successful artists don't think the lawsuits will benefit musicians.</td>
<td>even successful artists don't think the lawsuits will benefit musicians.</td>
</tr>
</tbody>
</table>

Table 2. Example inputs, outputs and references, highlighting the strengths and weaknesses of the Pointer-Generator model.

Future Work

Data: build dataset with other types of fluff.

Model experimentation: train and test a wider variety of summarization models.

References


Discussion

Extractive
- Identifies large portion of "fluff" expressions.
- Removes most expressions, but keeps chopped versions of some: ex. which is quite significant becomes which is and for all intents and purposes becomes for.
- Output of the text is a little sloppy (no punctuation) but does a decent job for our specific problem overall.
- LSTM units and the maximum output length were key parameters to define (experiment more with these if had more time).
- Model training is very slow due to the length of input and outputs.

Pointer Generator
- Identifies a large portion of "fluff" expressions.
- Yet, still keeps some.
- Coverage system fixed repetition issues, but not entirely. Lots of repetition in short passages. Model may assume minimum length.
- Occasionally removed information from the input.
- Model training is very slow due to length of input/output (in fig 1, only the steps in the red box were our dataset — the other ones were from the pretrained model).
**Motivation**

- People write every day.
- Text summarization is heavily explored.
- Getting rid of "fluff" (clutter in language)

**Goal**

- Given a textual input, output a concise version of that text (i.e. keep the same information in fewer words).

**Data**

- Approach
  - No pre-built dataset mapping cluttered text to concise text
  - We built our own
  - Reverse-engineered from concise to fluffed
    - Downloaded BBC dataset of news articles
    - Used a Text Inflator to add "fluff"
    - Original version => output
    - Fluffled version => input

**Stats**

- Fluffed articles: avg. of 523 words (min. 122, max. 6,208)
- Concise articles: avg of 384 words (min. 89, max. 4,342)

**Split**

- The 2,225 were divided as follows:
  - Train: 1,778 examples (80%)
  - Development: 223 examples (10%)
  - Test: 223 examples (10%)

**Models**

- **Basic LSTM (Extractive Approach)**
  - Inputs embedded fluffed text document with its associated binary labels on the concise version of the text
  - Outputs predictions for whether or not a given word in the fluffed document should appear in the concise version
  - Probability that a word is labeled to be in the output text based on model predictions: P = 0.95
  - Based on the binary labels generated, the predicted concise text is generated and compared to the ground truth concise text using ROUGE score.

- **Pointer Generator (Abstractive Approach)**
  - Pointing: copies words from original
  - Generator: creates new sentences (seq2seq + attn)
  - Bellow, p_gen is the probability of generating a word from the vocabulary (as opposed to copying).
  - Coverage: tracks content covered, avoids repetition.
  - Loss: finally, the loss combines the coverage loss with the original loss.

**Results**

- **Table 1. ROUGE scores on different models.**

**Discussion**

- **Extractive**
  - Identifies large portion of "fluff" expressions
  - Removes most expressions, but keeps chopped versions of some: **ex. which is quite significant becomes which is**
  - Output of the text is a little sloppy, (no punctuation) but does a decent job for our specific problem overall

- **Abstractive**
  - Identifies a large portion of "fluff" expressions.
  - Yet, still keeps some.
  - Coverage system fixed repetition issues, but not entirely. Lots of repetition in short passages. Model may assume minimum length.
  - Occasionally removed information from the input.
  - Model training is very slow due to length of input/output (in fig 1, only the steps in the teal box were in our dataset – the other ones were from the pretrained model).

**Future Work**

- Explore ways to deal with longer outputs without incurring huge time costs.

**References**

Cutting Down “Fluff”: A Twist on Text Summarization
Kate Salmon (ksalmon1@stanford.edu), Gustavo Torres da Silva (gdasilva@stanford.edu)

Motivation
- People write every day.
- Getting rid of language clutter takes time.
- Text summarization is heavily explored, but not as an a way to assist human writing.
- Goal: given a textual input, output a concise version of that text keeping the same information in fewer words.

Methods
- No pre-built dataset mapping cluttered text to concise text
- So we built our own
- Reverse-engineered from concise to fluffed:
  - Downloaded BBC dataset of news articles
  - Used a Text Inflator to add "fluff" to it
  - Original version => output
  - Fluffed version => input

Results

<table>
<thead>
<tr>
<th>Models</th>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>F-Score</td>
</tr>
<tr>
<td>Baseline</td>
<td>0.8134</td>
<td>0.2201</td>
<td>0.3383</td>
</tr>
<tr>
<td>PG</td>
<td>0.6067</td>
<td>0.7549</td>
<td>0.6089</td>
</tr>
<tr>
<td>PG + Cov</td>
<td>0.9498</td>
<td>0.7492</td>
<td>0.8185</td>
</tr>
</tbody>
</table>

References
- People write every day.
- Getting rid of language clutter takes time.
- Text summarization is heavily explored, but not as an a way to assist human writing.
- Goal: given a textual input, output a concise version of that text keeping the same information in fewer words.

Future Work
- People write every day.
- Getting rid of language clutter takes time.
- Text summarization is heavily explored, but not as an a way to assist human writing.
- Goal: given a textual input, output a concise version of that text keeping the same information in fewer words.
Motivation

- Wide variety of applications from gathering insights on climate change to analyzing economic development
- Goal: predict environmental changes utilizing satellite images in a time sequence

Problem Definition

For window size $t$, train a model $f$ to produce an output image $x_{t+1}$, that estimates the true $(t+1)$th image $x_{t+1}^0$:

$$f : x_{t} = \{x_{t}^{(1)}, x_{t}^{(2)}, \ldots, x_{t}^{(m)}\} \rightarrow x_{t+1}^{(i)}$$

Baseline

Given a temporal sequence of satellite images, we compute the images’ mean and use this as the prediction for the next timestamp’s image.

Pix2pix

Pix2pix uses conditional adversarial networks to implement image-to-image translation. We have:
- the generator $G$, that tries to generate an image that looks similar to images from the target domain $y$;
- the discriminator $D$, that learns to distinguish fake examples generated by $G$ from real ones in $y$.

Loss function:

$$(G, D) = \mathbb{E}_{x,y} \left[ \log D(x,y) + \right]
\mathbb{E}_{z \sim p_z} \left[ \log (1 - D(x, G(x,z))) \right]$$

We aim to solve:

$$G^* = \arg \min_G \max_D L(G,D)$$

Models

Architecture

We use the original Pix2pix architecture, whose modules are all of the form Convolution-Batchnorm-ReLU.

Results

<table>
<thead>
<tr>
<th>Models</th>
<th>Mean SSIM (val)</th>
<th>Mean SSIM (test)</th>
<th>Mean PSNR (val)</th>
<th>Mean PSNR (test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline ($t=1$, Clear)</td>
<td>0.489</td>
<td>0.494</td>
<td>18.5</td>
<td>18.9</td>
</tr>
<tr>
<td>Baseline ($t=1$, Cloudy)</td>
<td>0.401</td>
<td>0.410</td>
<td>15.6</td>
<td>15.5</td>
</tr>
<tr>
<td>Baseline ($t=5$, Clear)</td>
<td>0.518</td>
<td>0.525</td>
<td>18.1</td>
<td>18.4</td>
</tr>
<tr>
<td>Baseline ($t=5$, Cloudy)</td>
<td>0.427</td>
<td>0.429</td>
<td>15.4</td>
<td>15.3</td>
</tr>
<tr>
<td>Pix2Pix ($t=1$, Clear)</td>
<td>0.521</td>
<td>0.534</td>
<td>19.7</td>
<td>20.2</td>
</tr>
<tr>
<td>Pix2Pix ($t=5$, Cloudy)</td>
<td>0.477</td>
<td>0.480</td>
<td>17.9</td>
<td>18.0</td>
</tr>
<tr>
<td>Baseline ($t=1$, Cloudy)</td>
<td>0.428</td>
<td>0.450</td>
<td>15.2</td>
<td>15.2</td>
</tr>
<tr>
<td>Baseline ($t=5$, Cloudy)</td>
<td>0.419</td>
<td>0.429</td>
<td>14.8</td>
<td>14.6</td>
</tr>
<tr>
<td>Pix2Pix ($t=1$, Clear)</td>
<td>0.429</td>
<td>0.447</td>
<td>16.7</td>
<td>17.2</td>
</tr>
<tr>
<td>Pix2Pix ($t=5$, Cloudy)</td>
<td>0.462</td>
<td>0.469</td>
<td>17.6</td>
<td>17.3</td>
</tr>
</tbody>
</table>

Notice: Differences between SSIM/PSNR val/test results are most significant for $t=1$.

Discussion

Observations

- Even a simple baseline performs well, likely due to the staticity of landforms and the high temporal resolution
- Pix2pix outperforms the baseline for cloudy and non-cloudy images by mean SSIM and PSNR
- The wide performance gap between Pix2pix and baseline on cloudy data implies that GANs are able to capture more challenging mapping from input to output
- Tradeoff with window size
  - Baseline: large window sizes may result in excess noise (see right)
  - Main weakness of the models: limited ability to predict the amount and structure of cloud coverage in examples with volatile weather
  - Distribution of SSIM scores for the Pix2pix model trained on the non-cloudy dataset is more right-skewed than the distribution of the Pix2pix model trained on the cloudy dataset (see below)
  - Short-term weather (such as extra cloud coverage) cannot always be safely extrapolated to future predictions (see below)
- Many Pix2pix-generated images look qualitatively similar to the target; however, the SSIM for these images can still be low
- Visual structure of the generated clouds heavily impacts the SSIM

Contributions

- Applied GANs to challenging examples (e.g., volatile weather) and achieved results comparable to recent literature [1]
- Created a paired dataset of low-resolution images, since hi-resolution images are expensive and not publicly available

Future Work

- Collect more data across a wider variety of lat/lon/time.
- Temporal resolution: Prediction across larger time intervals should be less susceptible to volatile and short-term weather patterns and predict larger geological trends.
- Pix2pix with 3D convolutions: Explicitly incorporate temporal information.

References