OVERVIEW

- Related Work
- Problem Formulation
- Training Data for Summarization
- Reminder: Convolution Layer + LSTM
- Neural Summarization Model
  - Document Reader
  - Sentence Extractor
  - Word Extractor
- Implementation Details
- Results
- Conclusions
- Discussion
RELATED WORK

- Most extractive summarization methods relied on human-engineered features.
  - Surface (Radev et al., 2004), content (Nenkova et al., 2006), event (Filatova and Hatzivassiloglou, 2004) features
  - Score assigned to sentence

- Selection of sentences to be the summarization
  - binary classifiers (Kupiec et al., 1995)
  - hidden Markov models (Conroy and O’Leary, 2001)
  - graph-based algorithms (Erkan and Radev, 2004; Mihalcea, 2005)
  - integer linear programming (Woodsend and Lapata, 2010)
RELATED WORK

- neural network architectures for NLP
  - machine translation (Sutskever et al., 2014)
  - question answering (Hermann et al., 2015)
  - sentence compression (Rush et al., 2015)

- encoder-decoder architecture

- attention mechanism (Bahdanau et al., 2015)
  - introduced for translation
  - weighted combination of the input
PROBLEM FORMULATION

- Summaries at sentences level (sentence extraction)

\[ \log p(y_L | D; \theta) = \sum_{i=1}^{m} \log p(y^i_L | D; \theta) \]

- scoring each sentence within \( D \) (the source document)
- predicting a label \( y_L \in \{0,1\} \)
- \( \theta \) are the model parameters
- \( m \) is the number of sentences in \( D \)

+ little linguistic analysis for naturally grammatical summaries required
- long summaries containing much redundant information
PROBLEM FORMULATION

- Summaries at word level (word extraction)

$$\log p(y_s | D; \theta) = \sum_{i=1}^{k} \log p(w'_i | D, w'_1, \cdots, w'_{i-1}; \theta)$$

- $y_s = (w'_1, \cdots, w'_k), w'_i \in D$
- subset of words (can also include a small set of commonly-used words) in $D$ and their optimal ordering
TRAINING DATA

- Problem: large training corpus with labels needed
  - DUC 2002 (567 documents) only for testing

- DailyMail dataset (Hermann et al., 2015)
  - sentence extraction (200K articles)
    - rule-based algorithms to match highlights to document content
    - position of the sentence in the document
    - the unigram and bigram overlap between sentences and highlights
    - the number of entities appearing in the highlight and in the sentence

- word extraction (170K articles)
  - check if all highlight words (after stemming) come from the original document
  - out-of-vocabulary words, check for semantically equivalent replacement in the article (word2vec-GoogleNews-300dim-vectors)
REMINDER: CONVOLUTION LAYER

- used in image recognition/classifications
- maintains the relationship between pixels by using small squares of input data to learn features
  
  - **Kernel:**
    - learnable filter
  
  - **Stride:**
    - number of pixels over the input matrix

(Dumoulin et al., 2016)
REMINDER: LSTM

- vanilla RNN
  - vanishing/exploding gradient problem

- Long short-term memory (LSTM) (Schmidhuber, 1997)
  - minimizes vanishing/exploding gradient problem
REMINDER: LSTM

sigmoid  tanh  pointwise multiplication  pointwise addition  vector concatenation
REMINDER: LSTM

- Forget gate (what is relevant to keep from prior steps)

(Nguyen, 2018)
REMINDER: LSTM

- Input Gate (what is relevant to add from the current step)
REMINDER: LSTM

- Cell State (transfers relative information, “memory”)

\[
C_t = f_t \cdot C_{t-1} + i_t \cdot \hat{C}_t
\]
REMINDER: LSTM

- Output Gate (determines what the next hidden state should be)

(Nguyen, 2018)
NEURAL SUMMARIZATION MODEL

- Document Reader
- Convolutional Sentence Encoder
  - multiple kernels with different widths \{1, 2, 3, 4, 5, 6, 7\}
  - max-over-time pooling operation
  - summed to get the final sentence representation
Some random words in a sentence.
Some random words in a sentence.
Some random words in a sentence.
Some random words in a sentence.
NEURAL SUMMARIZATION MODEL

- Document Reader
- Recurrent Document Encoder
  - representations for documents using LSTMs
  - ameliorating the vanishing gradient problem when training long sequences
  - sequence of sentence vectors into a document vector
RECURRENT DOCUMENT ENCODER

Some random words in a sentence.
RECURRENT DOCUMENT ENCODER

Some random words in a sentence.
RECURRENT DOCUMENT ENCODER

Some random words in a sentence.
SENTENCE EXTRACTOR

- recurrent neural network labels sentences sequentially
  - “attention” applied directly to extract sentences
- labeling decision is made with both
  - encoded document at timestep $t$
  - previously labeled sentences $t - 1$
\[
\bar{h}_t = \text{LSTM}(p_{t-1}s_{t-1}, \bar{h}_{t-1}) \\
p(y_L(t) = 1|D) = \sigma(\text{MLP}(\bar{h}_t : h_t))
\]

- \(\bar{h}\) extractor hidden state
- \(p_t\) degree to which the last cell believes the previous sentence should be extracted and memorized
  - curriculum learning strategy
- \(h\) encoder hidden state
- \(\text{MLP}\) multi layer neural network
SENTENCE EXTRACTOR

Some random words in a sentence.
SENTENCE EXTRACTOR

Some random words in a sentence.
Some random words in a sentence.
Can be seen as a generation task
- words must be selected
- sentences rendered fluently and grammatically correct

Hierarchical attention architecture:
- decoder softly attends each document sentence
- subsequently attends each word in the document and computes the probability of the next word to be included in the summary

n-gram features collected from the document to rerank candidate summaries obtained via beam decoding
- log-linear reranker (Och, 2003)
Some random words in a sentence.
\[ \overline{h_t} = \text{LSTM}(w'_{t-1}, \overline{h}_{t-1}) \]

Some random words in a sentence.
\[ \bar{h}_t = \text{LSTM}(w'_{t-1}, \bar{h}_{t-1}) \]

\[ a^t_j = z^T \tanh(W_c \bar{h}_t + W_r h_j), h_j \in D \]

\[ b^t_j = \text{softmax}(a^t_j) \]
\[ \tilde{h}_t = \text{LSTM}(w'_{t-1}, \tilde{h}_{t-1}) \]

\[ a^t_j = z^T \tanh(W_c \tilde{h}_t + W_r h_j), h_j \in D \]

\[ b^t_j = \text{softmax}(a^t_j) \]

\[ \tilde{h}_t = \sum_{j=1}^{m} b^t_j h_j \]

\[ u^t_i = v^T \tanh(W_c' \tilde{h}_t + W_r' w_i), w_i \in D \]

\[ p(w'_t = w_i | D, w'_1, \cdots, w'_{t-1}) = \text{softmax}(u^t_i) \]
IMPLEMENTATION DETAILS

- input documents are padded to the same length
- size of word (= 150), sentence (=300), and document (= 750) embeddings
  - word vectors 150 dimensional word2vec pre-trained on Google 1-billion word benchmark (Chelba et al., 2014)
- convolutional kernel sizes \{1, 2, 3, 4, 5, 6, 7\}
- dropout with probability 0.5 on
  - the LSTM input-to-hidden layers
  - the scoring layer
IMPLEMENTATION DETAILS

- **Sentence Extractor**
  - number of sentences being selected (three sentences)
    - reranking the positively labeled sentences with the probability scores
    - obtained from the softmax layer (rather than the label itself)

- **Word Extractor**
  - negative sampling
    - vocabulary of different documents trimmed to the same length
RESULTS

- ROUGE (Lin and Hovy, 2003)
  - ROUGE -1,2 (unigram and bigram overlap)
    - assessing informativeness
  - ROUGE -L (longest common subsequence)
    - assessing fluency

- human judgments for ranking 20 randomly sampled DUC 2002 test documents on formativeness and fluency
  - sentence-based extraction
  - word-based extraction
  - neural abstractive system (Rush et al., 2015)
  - lead baseline (first three sentences)
  - phrase-based ILP model (Woodsend and Lapata, 2010)
  - human authored summary

- TGRAPH and URANK (DUC 2002 only)
## RESULTS

<table>
<thead>
<tr>
<th>DUC 2002</th>
<th>ROUGE -1</th>
<th>ROUGE -2</th>
<th>ROUGE -L</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEAD</td>
<td>43.6</td>
<td>21.0</td>
<td>40.2</td>
</tr>
<tr>
<td>LREG</td>
<td>43.8</td>
<td>20.7</td>
<td>40.3</td>
</tr>
<tr>
<td>ILP</td>
<td>45.4</td>
<td>21.3</td>
<td>42.8</td>
</tr>
<tr>
<td>NN - ABS</td>
<td>15.8</td>
<td>5.2</td>
<td>13.8</td>
</tr>
<tr>
<td>TGRAPH</td>
<td>48.1</td>
<td>24.3</td>
<td>-</td>
</tr>
<tr>
<td>URANK</td>
<td>48.5</td>
<td>21.5</td>
<td>-</td>
</tr>
<tr>
<td>NN - SE</td>
<td>47.4</td>
<td>23.0</td>
<td>43.5</td>
</tr>
<tr>
<td>NN - WE</td>
<td>27.0</td>
<td>7.9</td>
<td>22.8</td>
</tr>
</tbody>
</table>
## RESULTS

<table>
<thead>
<tr>
<th>DailyMail</th>
<th>ROUGE -1</th>
<th>ROUGE -2</th>
<th>ROUGE -L</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEAD</td>
<td>20.4</td>
<td>7.7</td>
<td>11.4</td>
</tr>
<tr>
<td>LREG</td>
<td>18.5</td>
<td>6.9</td>
<td>10.2</td>
</tr>
<tr>
<td>NN - ABS</td>
<td>7.8</td>
<td>1.7</td>
<td>7.1</td>
</tr>
<tr>
<td>NN - SE</td>
<td>21.2</td>
<td>8.3</td>
<td>12.0</td>
</tr>
<tr>
<td>NN - WE</td>
<td>15.7</td>
<td>6.4</td>
<td>9.8</td>
</tr>
</tbody>
</table>
## RESULTS

<table>
<thead>
<tr>
<th></th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
<th>6th</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEAD</td>
<td>0.10</td>
<td>0.17</td>
<td>0.37</td>
<td>0.15</td>
<td>0.16</td>
<td>0.05</td>
<td>3.27</td>
</tr>
<tr>
<td>ILP</td>
<td>0.19</td>
<td>0.38</td>
<td>0.13</td>
<td>0.13</td>
<td>0.11</td>
<td>0.06</td>
<td>2.77</td>
</tr>
<tr>
<td>NN-ABS</td>
<td>0.00</td>
<td>0.01</td>
<td>0.05</td>
<td>0.16</td>
<td>0.23</td>
<td>0.54</td>
<td>5.24</td>
</tr>
<tr>
<td>NN-SE</td>
<td>0.22</td>
<td>0.28</td>
<td>0.21</td>
<td>0.14</td>
<td>0.12</td>
<td>0.03</td>
<td>2.74</td>
</tr>
<tr>
<td>NN-WE</td>
<td>0.00</td>
<td>0.04</td>
<td>0.03</td>
<td>0.21</td>
<td>0.51</td>
<td>0.20</td>
<td>4.79</td>
</tr>
<tr>
<td>Human</td>
<td>0.27</td>
<td>0.23</td>
<td>0.29</td>
<td>0.17</td>
<td>0.03</td>
<td>0.01</td>
<td>2.51</td>
</tr>
</tbody>
</table>
CONCLUSIONS

- data-driven summarization framework based on an encoder-extractor architecture.
- interesting comparison to human summaries
- Word Extractor has interesting architecture with a reranker rather than just looking at the combined likelihood
  - reranker not described
- for conv layer, rather stack small kernels than using big ones
  - $7 \times 7 \iff 3 \times (3 \times 3)$ but less weights + more nonlinearity

- rules for data labeling as well as dataset was not published
- “attention” for sentence extractor only takes one sentence of the document into account
THANK YOU


Michael Nguyen. 2018. Illustrated Guide to LSTM’s and GRU’s: A step by step explanation


Vincent Dumoulin, Francesco Visin. 2016. A guide to convolution arithmetic for deep learning
DISCUSSION

• Why do Cheng et al. call the concatenation of encoder and decoder state "attention"?

\[ p(y_L(t) = 1|D) = \sigma(\text{MLP}(\tilde{h}_t : h_t)) \]

• Nallapati et al. argue, Cheng et al. needed higher annotation costs, because they used manually created labels. But did they not instead created training sets via rules and heuristics. Isn't that at least a semi-automatic approach?
  • rules take into account
    • the position of the sentence
    • the unigram and bigram overlap between sentences and highlights
    • the number of entities appearing in the highlight and in the sentence
  • “We adjusted the weights of the rules on 9,000 documents with manual sentence labels created by Woodsend and Lapata (2010).”
DISCUSSION

- It sounds like their word extraction model is abstractive but with the vocabulary fixed to words in the document.
  - “conditional language model with a vocabulary constraint”
    - assigns probabilities to a sequence of words given some context
    - constraint is vocabulary of the document
      - can also be extended to include a small set of commonly-used (high-frequency) words

- Do you think that the combined architecture of CNN+LSTM performs a lot better than considering a single paragraph vector and sentence vector concept based on word2vec/fasttext models?
    - better results for languages with rich morphology
  - For Encoding important for long sentences as initial information might get lost otherwise
  - LSTM allows model to know what sentence have been extracted before to not extract a sentence with redundant information
THANK YOU