IRE MAJOR PROJECT

Neural Summarization of documents by extracting sentences

Submitted by:

Group8
Suryansh Agnihotri
Anandita Dhasmana
Piyush Shrivastava
Shalin Agarwal

Abstract

Traditional approaches to extractive summarization rely heavily on human-engineered features.We have used a data-driven approach based on neural networks and continuous sentence features.The frame-work for single-document summarization composed of a hierarchical document encoder and an attention-based extractor.This architecture allows us to develop different classes of summarization models which can extract sentences or words. We train our models on large scale corpora containing hundreds of thousands of document-summary pairs.

Introduction

The need to access and digest large amounts of textual data has provided strong impetus to develop automatic summarization systems aiming to create shorter versions of one or more documents, whilst preserving their information content. Much effort in automatic summarization has been devoted to sentence extraction, where a summary is created by identifying and subsequently concatenating the most salient text units in a document.

Most extractive methods to date identify sentences based on human-engineered features.These include surface features such as sentence position and length , the words in the title, the presence of proper nouns, content features such as word frequency, and event features such as action nouns.

Sentences are typically assigned a score indicating the strength of presence of these features.

We used a data-driven approach for summarization based on neural networks and continuous sentence features.

We develop a general framework for single-document summarization which can be used to extract sentences or words. Our model includes a neural network-based hierarchical document reader or encoder and an attention-based content extractor. The role of the reader is to derive the meaning representation of a document based on its sentences and their constituent words. Our models adopt a variant of neural attention to extract sentences. Our model applies attention directly to select sentences of the input document as the output summary.

Our decoder selects output symbols from the document of interest rather than the entire vocabulary. This effectively helps us sidestep the difficulty of searching for the next output symbol under a large vocabulary, with low-frequency words and named entities whose representations can be challenging to learn.

Mathematical formulation of the problem

Given a document D consisting of a sequence of sentences $\{s_1, \cdots, s_m\}$ Sentence extraction aims to create a summary from D by selecting a subset of j sentences (where $j < m$). We do this by scoring each sentence within D and predicting a label $y_L \in \{0, 1\}$ indicating whether the sentence should be included in the summary.

Objective function to be maximized by supervised training:

The objective is to maximize the likelihood of all sentence labels:

$$y_L = (y_L^1, \cdots, y_L^m)$$

Given the input document D and model parameters θ:

$$\log p(y_L | D; \theta) = \Sigma \log p(y_L^i | D; \theta)$$

Training Data for Summarization

We used DailyMail dataset which was created using rule based system that determines whether a document sentence matches a highlight and should be labeled with 1 (must be in the summary), and 0 otherwise. The rules take into account the position of the sentence in the document, the unigram and bigram overlap between document sentences and highlights, the number of entities appearing in the highlight and in the document sentence.

Dataset division:

90% of data set for training

5% for validation

5%for testing

Example doc in daily mail dataset:

```
as the model for @entity4 's " @entity3 , " @entity1 became the symbol of @entity7 women working on the home front during @entity9 the 92 - year - old died this week
at her home in @entity12 , @entity13                    1
as a 19 - year - old telephone operator , @entity1 posed for the famous painting that would become the cover of the @entity17 on may 29 , 1943                 1
although she was petite , @entity1 was transformed into the iconic -- and burly -- embodiment of the character by @entity4                      1
" other than the red hair and my face , @entity4 embellished @entity26 's body , " @entity1 said in a 2012 interview with the @entity22            1
" i was much smaller than that and did not know how he was going to make me look like that until i saw the finished painting                    0
" people we 've lost in 2015 @entity1 pocketed $ 10 for the two mornings of modeling work she did in @entity34 , @entity35                     2
@entity4 lived in neighboring @entity34 at the time                     0
" @entity3 " is often confused with another popular image from the same era                  2
the poster shows a woman flexing her arm under the slogan " @entity42                  0
" it was part of a nationwide campaign to sell war bonds , but is not the same character                        0
still , many folks on social media paid tribute to @entity1 using the image                 2
both show the key role women played in the war effort .                 0

" @entity3 " appeared on the cover of the @entity17 on may 29 , 1943
@entity1 was a 19 - year - old telephone operator at the time

@entity3:Rosie the Riveter
@entity17:Saturday Evening Post
@entity1:Mary Doyle Keefe
@entity0:CNN
```

Neural Summarization Model

The key components of our summarization model include a neural network-based hierarchical document reader and an attention-based hierarchical content extractor. The hierarchical nature of our model reflects the intuition that documents are generated compositionally from words, sentences, paragraphs, or even larger units. We therefore employ a representation framework which reflects the same architecture, with global information being discovered and local information being preserved.

Such a representation yields minimum information loss and is flexible allowing us to apply neural attention for selecting salient sentences.

Key components used:

-neural network-based hierarchical document reader

-attention -based hierarchical content extractor

Document Reader

The role of the reader is to derive the meaning representation of the document from its constituent sentences, each of which is treated as a sequence of words.

-Convolutional Sentence Encoder

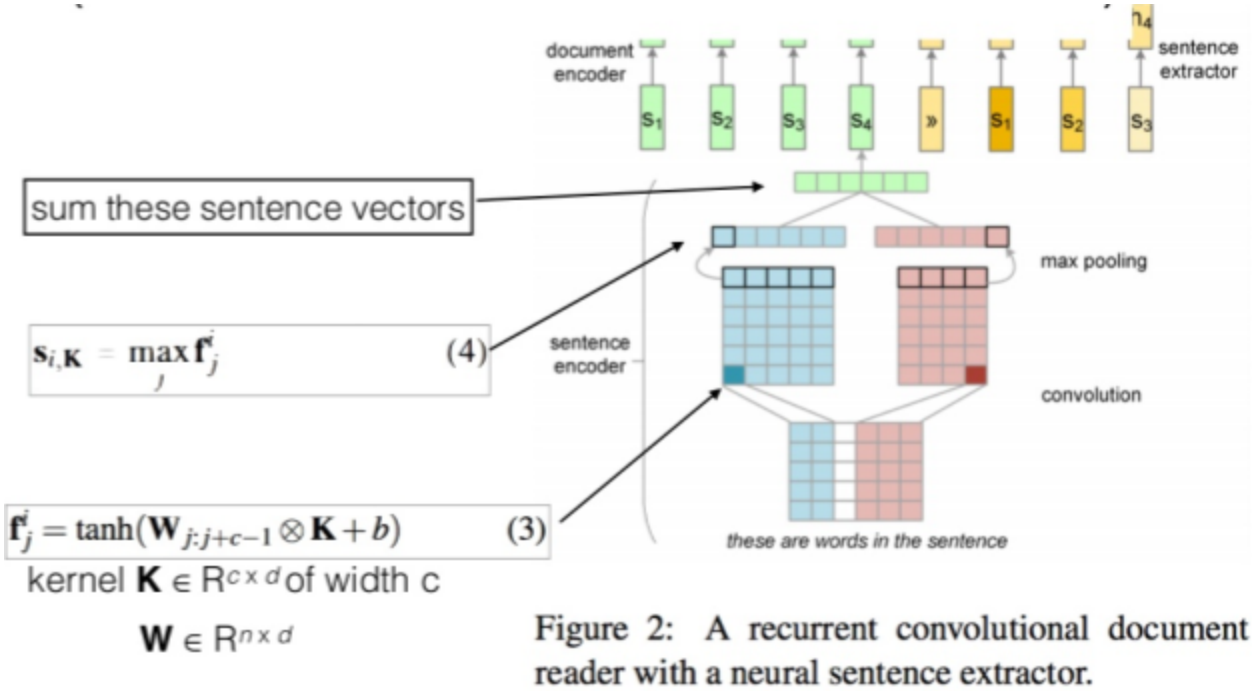
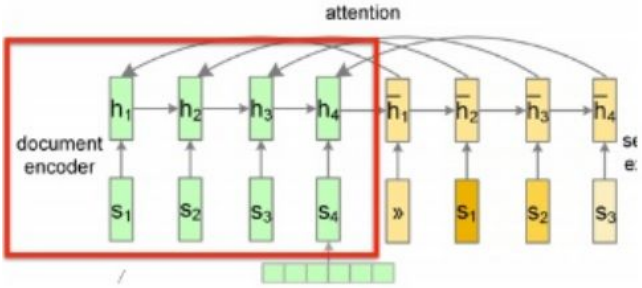


Figure 2: A recurrent convolutional document reader with a neural sentence extractor.

-Recurrent Document Encoder

Long Short-Term Memory (LSTM) activation unit is used in the RNN for ameliorating the vanishing gradient problem when training long sequences



$$\begin{bmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \hat{\mathbf{c}}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \mathbf{W} \cdot \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{s}_t \end{bmatrix}$$

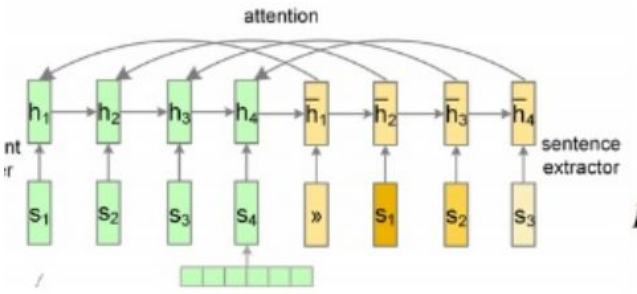
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$

Sentence Extractor(Document Decoder with attention mechanism)

Sentence extractor applies attention to directly extract salient sentences after reading them. It is another recurrent neural network that labels sentences sequentially, taking into account not only whether they are individually relevant but also mutually redundant.The next labeling decision is made with both the encoded document and the previously labeled sentences in mind.

Given encoder hidden states ($h_1, \cdots, h_m$) and extractor hidden states ($\bar{h}_1, \cdots, \bar{h}_m$) at time step t, the decoder attends the t-th sentence by relating its current decoding state to the corresponding encoding state shown in the eq.:



$$\bar{\mathbf{h}}_t = \mathrm{LSTM}(p_{t-1}\mathbf{s}_{t-1}, \bar{\mathbf{h}}_{t-1}) \quad (8)$$

$$p(y_L(t) = 1|D) = \sigma(\mathrm{MLP}(\bar{\mathbf{h}}_t : \mathbf{h}_t)) \quad (9)$$

10

At the beginning , set p_{t-1} to the true label of the previous sentence;
as training goes on , gradually shift its value to the predicted label :p_{t-1} => $p_{predicted}$

Results

Training results:

Model was trained in 18 epoch of the training dataset:

```
group8@kattapa: ~                                                    En  (47%)  1:49 PM
Terminal                                          group8@kattapa: ~
79279: 18 [ 4075/ 4178], train_loss/perplexity = 0.72399080/2.0626485 secs/batch = 1.0302s, grad.norm=3.57964730
79284: 18 [ 4080/ 4178], train_loss/perplexity = 0.71745735/2.0492160 secs/batch = 1.0298s, grad.norm=3.56070662
79289: 18 [ 4085/ 4178], train_loss/perplexity = 0.59847748/1.8193467 secs/batch = 0.9286s, grad.norm=3.61784101
79294: 18 [ 4090/ 4178], train_loss/perplexity = 0.65526217/1.9256473 secs/batch = 1.0238s, grad.norm=4.04992962
79299: 18 [ 4095/ 4178], train_loss/perplexity = 0.74765605/2.1120436 secs/batch = 0.9192s, grad.norm=3.78509855
79304: 18 [ 4100/ 4178], train_loss/perplexity = 0.64466041/1.9053398 secs/batch = 0.9187s, grad.norm=3.40437174
79309: 18 [ 4105/ 4178], train_loss/perplexity = 0.62093174/1.8606609 secs/batch = 0.9329s, grad.norm=3.28407669
79314: 18 [ 4110/ 4178], train_loss/perplexity = 0.68803763/1.9898070 secs/batch = 1.0285s, grad.norm=3.45331430
79319: 18 [ 4115/ 4178], train_loss/perplexity = 0.68428707/1.9823581 secs/batch = 1.0183s, grad.norm=4.28032923
79324: 18 [ 4120/ 4178], train_loss/perplexity = 0.64646292/1.9087774 secs/batch = 1.0554s, grad.norm=3.51886892
79329: 18 [ 4125/ 4178], train_loss/perplexity = 0.68874067/1.9912064 secs/batch = 1.0099s, grad.norm=3.67687058
79334: 18 [ 4130/ 4178], train_loss/perplexity = 0.63959587/1.8957146 secs/batch = 1.0251s, grad.norm=3.39978766
79339: 18 [ 4135/ 4178], train_loss/perplexity = 0.68236035/1.9785423 secs/batch = 1.0220s, grad.norm=3.64177942
79344: 18 [ 4140/ 4178], train_loss/perplexity = 0.64551520/1.9069693 secs/batch = 1.0247s, grad.norm=3.03482938
79349: 18 [ 4145/ 4178], train_loss/perplexity = 0.61482227/1.8493279 secs/batch = 1.0692s, grad.norm=3.68893623
79354: 18 [ 4150/ 4178], train_loss/perplexity = 0.67522871/1.9644822 secs/batch = 0.9729s, grad.norm=3.64155030
79359: 18 [ 4155/ 4178], train_loss/perplexity = 0.66250521/1.9396455 secs/batch = 0.9158s, grad.norm=3.31925321
79364: 18 [ 4160/ 4178], train_loss/perplexity = 0.62858111/1.8749484 secs/batch = 0.9124s, grad.norm=3.46081281
79369: 18 [ 4165/ 4178], train_loss/perplexity = 0.69106019/1.9958304 secs/batch = 0.9242s, grad.norm=3.65306425
79374: 18 [ 4170/ 4178], train_loss/perplexity = 0.66900069/1.9522854 secs/batch = 0.9221s, grad.norm=3.78807926
79379: 18 [ 4175/ 4178], train_loss/perplexity = 0.72488147/2.0644863 secs/batch = 1.0228s, grad.norm=3.53909063
Epoch training time: 4157.0442369
        > validation loss = 0.77011621, perplexity = 2.16001725
        > validation loss = 0.80609322, perplexity = 2.23914313
        > validation loss = 0.73013359, perplexity = 2.07535791
        > validation loss = 0.76211077, perplexity = 2.14279437
        > validation loss = 0.75037563, perplexity = 2.11779547
        > validation loss = 0.79909271, perplexity = 2.22352266
        > validation loss = 0.81238717, perplexity = 2.25328064
        > validation loss = 0.79477906, perplexity = 2.21395183
        > validation loss = 0.78517550, perplexity = 2.19279170
        > validation loss = 0.74409014, perplexity = 2.10452580
        > validation loss = 0.80345118, perplexity = 2.23323488
        > validation loss = 0.76084250, perplexity = 2.14007854
at the end of epoch: 18
train loss = 0.68240189, perplexity = 1.97862446
validation loss = 0.78737346, perplexity = 2.19761670
Saved model cv/epoch018_0.7874.model
validation perplexity did not improve enough, decay learning rate
learning rate was: 1.52588e-05
learning rate too small - stopping now
group8@kattapa:~/NeuralSum-master$
```

Score generation for Test data:

Sentence scores are generated and stored

```
9.949956494383513927e-01 7.101479917764663696e-01 7.060083001852035522e-01 1.238418444991111755e-01 4.624989330768585205e-01 1.993201747536659241e-01
3.840281814336776733e-01 7.199154645204544067e-01 2.584178000688552856e-01 3.626094162464141846e-01 5.996743440628051758e-01 1.217002160847187042e-01
5.549399703741073608e-01 3.994210436940193176e-01 1.621807896867721865e-07
9.803755236789584160e-01 9.883490633219480515e-01 9.616373460739850998e-01 4.649035930633544922e-01 8.931522220373153687e-01 9.598650056868791580e-01
8.513485267758369446e-02 4.951936900615692139e-01 6.704502403736114502e-01 4.238985180854797363e-01 2.506899759173393250e-01 4.282903745770454407e-01
1.948903873562812805e-01 1.339862719178199768e-01 5.209758654236793518e-01
9.470038060098886490e-01 3.329327851533889771e-01 8.667793422937393188e-01 3.271220773458480835e-01 8.187378868460655212e-01 9.322041794657707214e-01
3.561458736658096313e-01 9.533735625445842743e-01 9.108157195150852203e-01 8.284008875489234924e-01 1.474819257855415344e-01 2.700386419892311096e-01
7.888400629162788391e-01 2.834613621234893799e-01 8.629281520843505859e-01
9.740489851683378220e-01 9.155891425907611847e-01 3.406101912260055542e-01 2.958291769027709961e-01 5.060718208551406860e-01 4.430162720382213593e-02
5.825996696949005127e-01 3.519710898399353027e-01 5.402918159961700439e-01 4.129633903503417969e-01 4.437561333179473877e-01 4.655369818210601807e-01
8.468578308820724487e-01 8.318230584263801575e-01 8.672466501593589783e-02
9.913309137336909771e-01 9.209049306809902191e-01 9.734167642891407013e-01 8.478100746870040894e-01 2.522733360528945923e-01 3.865976333618164062e-01
9.415631815791130066e-01 5.875742435455322266e-01 8.117417097091674805e-01 1.138487346470355988e-01 7.902673035860061646e-01 1.874222531914710999e-01
3.138754889369010925e-01 5.088656246662139893e-01 4.671209752559661865e-01
```

Sentence scores are stored during evaluation.

	sentence1	sentence2				
doc1	8.42e-01	8.13e-01	8.37e-01	6.27e-01	5.07e-01	6.57e-01
doc2	9.12e-01	7.97e-01	8.03e-01	7.61e-01	8.85e-01	7.47e-01
	9.07e-01	8.68e-01	9.13e-01	9.46e-01	7.22e-01	9.21e-01

Each sentence of the document is labeled with probability of getting selected in the summary.

Conclusion and Future work:

We have used a data-driven summarization framework based on an encoder-decoder architecture for sequence labeling task.

This architecture can further be extended for generating summaries in english using the word extractor. Compared to sentence extraction which is a purely sequence labeling task, word extraction is closer to a generation task where relevant content must be selected and then rendered fluently and grammatically. A small extension to the structure of the sequential labeling model makes it suitable for generation: instead of predicting a label for the next sentence at each time step, the model directly outputs the next word in the summary.

References

1. Neural Summarization by Extracting Sentences and Words (https://arxiv.org/pdf/1603.07252.pdf)
2. http://colah.github.io/posts/2015-08-Understanding-LSTMs/