A Hierarchical Neural Autoencoder for Paragraphs and Documents

Akilesh B, CS13B1042

Indian Institute of Technology, Hyderabad

November 26, 2016

Introduction

- In this work, we explore an important step toward this generation task: training an LSTM (Long Short Term Memory) auto-encoder to preserve and reconstruct multi-sentence paragraphs.
- We focus on the component task of training a paragraph (document)-to-paragraph (document) autoencoder to reconstruct the input text sequence from a compressed vector representation from a deep learning model
- We develop hierarchical LSTM models that arranges tokens, sentences and paragraphs in a hierarchical structure, with different levels of LSTMs capturing compositionality at the token-to-token and sentence-to-sentence levels.

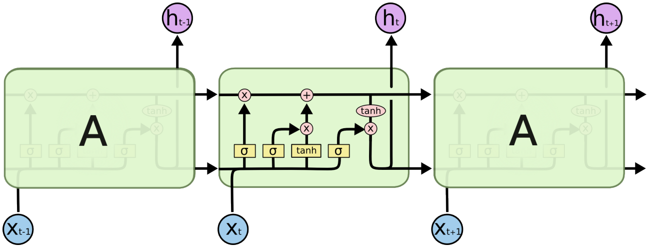
Intro contd . . .

- Recent LSTM models (Hochreiter and Schmidhuber, 1997) have shown powerful results on generating meaningful and grammatical sentences in sequence generation tasks like machine translation (Sutskever et al., 2014; Bahdanau et al., 2014; Luong et al., 2015) or parsing (Vinyals et al., 2014).
- This performance is at least partially attributable to the ability of these systems to capture local compositionally: the way neighboring words are combined semantically and syntactically to form meanings that they wish to express.

Motivation

- Could these models be extended to deal with generation of larger structures like paragraphs or even entire documents?
- In standard sequence to sequence generation tasks, an input sequence is mapped to a vector embedding that represents the sequence, and then to an output string of word. Multi-text generation tasks like summarization works in a similar way.
- Just as the local semantics of words can be captured by LSTM models, can the semantics of higher-level text units (e.g., clauses, sentences, paragraphs, and documents) be captured in a similar way.

LSTM



Approach

- In this paper, the target is to use an encoder-decoder model to regenerate the document.
- Specifically, a hierarchical LSTM model is used as an encoder, in which the representations of sentences are learned by a LSTM model whose inputs are words.
- The representation of the document is learned by another LSTM model whose inputs are sentences representations.
- A normal LSTM is used as a decoder.

Model 1 : Standard LSTM

- The whole input and output are treated as one sequence of tokens.
- Following Sutskever et al. (2014) and Bahdanau et al. (2014), we trained an autoencoder that first maps input documents into vector representations from a $LSTM_{encode}$ and then reconstructs inputs by predicting tokens within the document sequentially from a $LSTM_{decode}$.
- Two separate LSTMs are implemented for encoding and decoding with no sentence structures considered.

Standard Sequence to Sequence Model

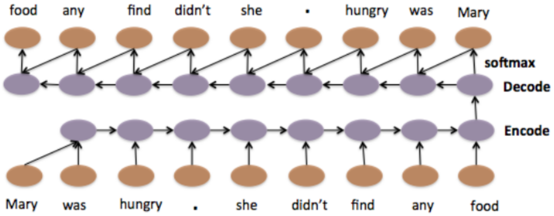


Figure 1: Standard Sequence to Sequence Model.

Model 2 : Hierarchical LSTM

Encoder We first obtain representation vectors at the sentence level by putting one layer of LSTM (denoted as $LSTM^{word}_{encode}$) on top of its containing words:

$$h^w_t(enc) = LSTM^{word}_{encode}(e^w_t, h^w_{t-1}(enc)) \tag{1}$$

The vector output at the ending time-step is used to represent the entire sentence as $e_s = h^w_{end_s}$

Hierarchical LSTM contd. . . .

To build representation e_D for the current document/paragraph D, another layer of LSTM (denoted as $LSTM^{sentence}_{encode}$) is placed on top of all sentences, computing representations sequentially for each time step:

$$h^s_t(enc) = LSTM^{sentence}_{encode}(e^s_t, h^s_{t-1}(enc)) \qquad (2)$$

Representation $e^s_{end_D}$ computed at the final time step is used to represent the entire document: $e_D = h^s_{end_D}$

Thus one LSTM operates at the token level, leading to the acquisition of sentence-level representations that are then used as inputs into the second LSTM that acquires document-level representations, in a hierarchical structure.

Hierarchical LSTM contd. . . .

Decoder As with encoding, the decoding algorithm operates on a hierarchical structure with two layers of LSTMs. LSTM outputs at sentence level for time step t are obtained by:

$$h_t^s(dec) = LSTM_{decode}^{sentence}(e_t^s, h_{t-1}^s(dec)) \quad (3)$$

The initial time step $h_0^s(d) = e_D$, the end-to-end output from the encoding procedure, $h_t^s(d)$ is used as the original input into the $LSTM_{decode}^{word}$ for subsequently predicting tokens within sentence $t+1$.

$LSTM_{decode}^{word}$ predicts tokens at each position sequentially, the embedding of which is then combined with earlier hidden vectors for the next time step prediction until the end_s token is predicted.

Hierarchical Sequence to Sequence Model

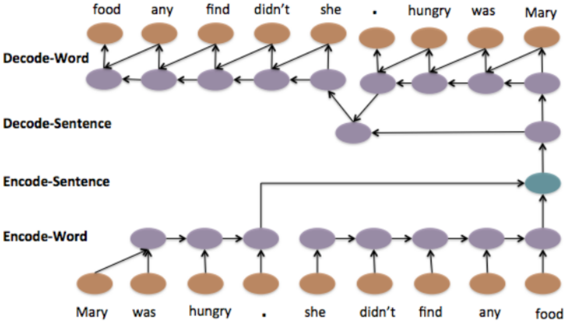


Figure 2: Hierarchical Sequence to Sequence Model.

Dataset

- **Hotel Reviews**: The reviews consisting sentences ranging from 50 to 250 words are considered. The vocabulary set consisted of the 25,000 most frequent words. A special UNK token is used to denote all the less frequent tokens. Reviews that consist of more than 2 percent of unknown words are discarded. The training dataset is comprised of 340,000 reviews; the testing set is comprised of 40,000 reviews.
- **Wikipedia**:Paragraphs are extracted from Wikipedia corpus that meet the aforementioned length requirements. Paragraphs with larger than 4 percent of unknown words are discarded. The training dataset is comprised of roughly 500,000 paragraphs and testing contains roughly 50,000.

Training and Implementation Details

- LSTM parameters and word embeddings are initialized from a uniform distribution between [-0.08, 0.08].
- Each LSTM layer consists of 1000 hidden neurons and the dimensionality of word embeddings is set to 1000
- Decoding algorithm allows generating at most 1.5 times the number of words in inputs.

Evaluation Metric

We use ROUGE-1,ROUGE-2 scores on the dataset.
ROUGE is a recall-oriented measure widely used in the language generation literature. It measures the n-gram recall between the candidate text and the reference (generated) text(s).

$$ROUGEn = \frac{\sum_{gram_n \in input} count_{match}(gram_n)}{\sum_{gram_n \in input} count(gram_n)} \qquad (4)$$

where $count_{match}$ denotes the number of n-grams co-occurring in the input and output.
ROUGE How much the words (and/or n-grams) in the human reference text appeared in the machine generated text. It meassures recall.

Evaluation Metric . . .

BLEU Purely measuring recall will inappropriately reward long outputs. BLEU is designed to address such an issue by emphasizing precision. n-gram precision scores for our situation are given by:

$$precision_n = \frac{\sum_{gram_n \in output} count_{match}(gram_n)}{\sum_{gram_n \in output} count(gram_n)} \tag{5}$$

BLEU How much the words (and/or n-grams) in the machine generated text appeared in the human reference text. It measures precision.

Results

Table: Results for the LSTM Models

Model	Dataset	BLEU	ROUGE-1	ROUGE-2
Standard	Hotel Review	0.241	0.571	0.302
Standard	Wikipedia	0.178	0.502	0.228
Hierarchical	Hotel Review	0.267	0.590	0.330
Hierarchical	Wikipedia	0.202	0.529	0.250

Hierarchical LSTM examples

Input-Wiki:

paris is the capital and most populous city of france . situated on the seine river , in the north of the country , it is in the centre of the le-de-france region . the city of paris has a population of 2273305 inhabitants . this makes it the fifth largest city in the european union measured by the population within the city limits .

Output-Wiki:

paris is the capital and most populated city in france . located in the UNK , in the north of the country , it is the center of UNK . paris , the city has a population of NUM inhabitants . this makes the eu s population within the city limits of the fifth largest city in the measurement

Hierarchical LSTM examples . . .

Input-Review:

on every visit to nyc , the hotel beacon is the place we love to stay . so conveniently located to central park , lincoln center and great local restaurants . the rooms are lovely . beds so comfortable , a great little kitchen and new wizz bang coffee maker . the staff are so accommodating and just love walking across the street to the fairway supermarket with every imaginable goodies to eat .

Output-Review

every time in new york , lighthouse hotel is our favorite place to stay . very convenient , central park , lincoln center , and great restaurants . the room is wonderful , very comfortable bed , a kitchenette and a large explosion of coffee maker . the staff is so inclusive , just across the street to walk to the supermarket channel love with all kinds of what to eat .

Dataset

The Signal Media One-Million News Articles Dataset is used for training and testing. It contains 1 million articles that are mainly English, but they also include non-English and multi-lingual articles.

Each article in the dataset has the following fields.

- **id**: a unique identifier for the article.
- **title**: the title of the article.
- **content**: the textual content of the article (which occasionally contained HTML and JavaScript content).
- **source**: the name of the article source (e.g. Reuters).
- **published**: the publication date of the article.
- **media-type**: either "News" or "Blog".

Data Pre-processing

- Minimal pre-processing such as lower-casing, separating on the basis of punctuation marks are applied.
- All the punctuations are removed in order to reduce the vocabulary size.
- We remove the articles with more than 20% label the tokens which occur less than 10 times in our corpus.
- For every article, cosine similarity of each sentence present in it, is computed with the corresponding title. We retain only the sentences with top 5 cosine similarity scores per article.
- The average number of words in an article is 122 .

Constructing feature vector and training

- We initially trained a Word2vec model on our whole corpus, which is a two-layer neural net that generates word embeddings. We found out that the results were not good.
- We used a one-hot encoding for our feature vector. This means the size of feature vector will be equal to the vocabulary.
- The standard LSTM encoder decoder model is implemented in Keras / Theano.
- Adagrad/Adadelta is used as optimizer for training.

Training details

Table: Hyperparameter list

Hyperparameter	Value
Hidden layer size	500/1000
Batch size	128
Learning rate	0.1
Optimizer	Adagrad/Adadelta

- The training was performed on a Tesla K-20 GPU with 6GB RAM.

Standard LSTM examples

Generated Text:
the new remote control about thicker than ray previous changed ipad apps of case girl s the good thing
Original Text:
the new remote control is thicker than the previous one and in this case that s a good thing
Generated Text: (Bad Example)
to we've difference and diversity we've been by
Original Text:
where we've succeeded, its because we've been different.

Hierarchical LSTM examples

Generated Text:
if we spread malicious gossip we our poisoning words own peace demoralise sabotaging situations own influence
Original Text:
if we spread malicious gossip we are poisoning our own future and sabotaging our own influence with others
Generated Text:
pressing down on the screen starts different actions depending fathers how much force healing exerted
Original Text:
pressing down on the screen starts different actions depending on how much force is exerted

Hierarchical LSTM examples . . .

Generated Text:
the misunderstood tried searching i action movies the comedy movies the the remote control
Original Text:
i also tried searching for action movies and comedy movies with the remote control s microphone
Generated Text:
bowling ball on third 3rd side with ease redzone left texas went to chris nutall
Original Text:
bowling ball on third 3rd and in the redzone texas state went to chris nutall senior running back

Results

Table: Results for the LSTM Models

Model	No of articles	BLEU	ROUGE-1	ROUGE-2
Standard	500	0.1267	0.4113	0.2591
Hierarchical	200	0.2372	0.8143	0.6014
Hierarchical	500	0.3513	0.7713	0.5945

Challenges faced

- Computational constraints - GPU.
- Huge size of vocabulary due to one-hot encoding.
- Hyperparameter optimizations which ensure error function converges to an appropriate local minima.

References

- A Hierarchical Neural Autoencoder for Paragraphs and Documents. Jiwei et al.
- Sequence to sequence learning with Neural Networks. Sutskever et al.
- http://research.signalmedia.co/newsir16/signal-dataset.html