DSME 6635: Artificial Intelligence for Business Research

Deep-Learning-based NLP: RNN and Seq2Seq

Renyu (Philip) Zhang

1

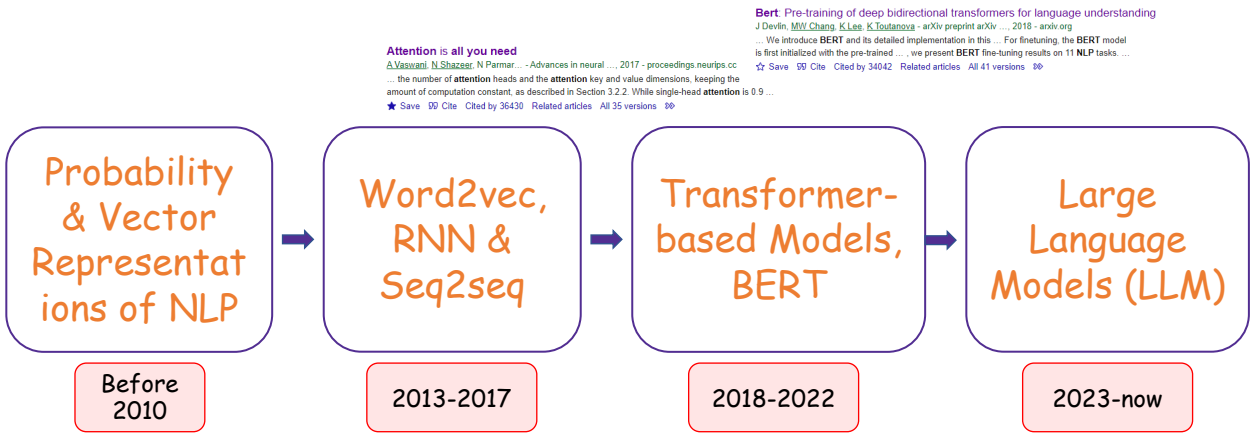
Agenda

- Vanilla Recurrent Neural Nets (RNN)
- Long Short-Term Memory (LSTM)
- Sequence-to-sequence (Seq2seq)

2

2

NLP Roadmap



3

3

Why Does Sequence Matter?

- Consider the following case:

This is not a real restaurant, it's a filthy burger joint.

This is not a filthy burger joint, it's a real restaurant.

- If we only use word embeddings for sentiment classification, these two opposite sentences will generate the same sentence vector (i.e., the sentence vector is the **sum** of the word vectors for all words in the sentence).

4

4

Back to N-Gram Models

- N-gram model is a **language model** that limits the dependencies on history, a.k.a. **Markov Chain**.

$$P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(1)}) = P(\boldsymbol{x}^{(t+1)}|\underbrace{\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)}}_{n\text{-1 words}})$$ (assumption)

prob of a n-gram / prob of a (n-1)-gram

$$= \frac{P(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{P(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}$$ (definition of conditional prob)

- **Question:** How do we get these *n*-gram and (*n*-1)-gram probabilities?
- **Answer:** By counting them in some large corpus of text!

$$\approx \frac{\text{count}(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{\text{count}(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}$$ (statistical approximation)

Two important issues:

- Sparsity (partially addressed by smoothing)
- Model Size (n is no more than 5, usually 2 or 3)

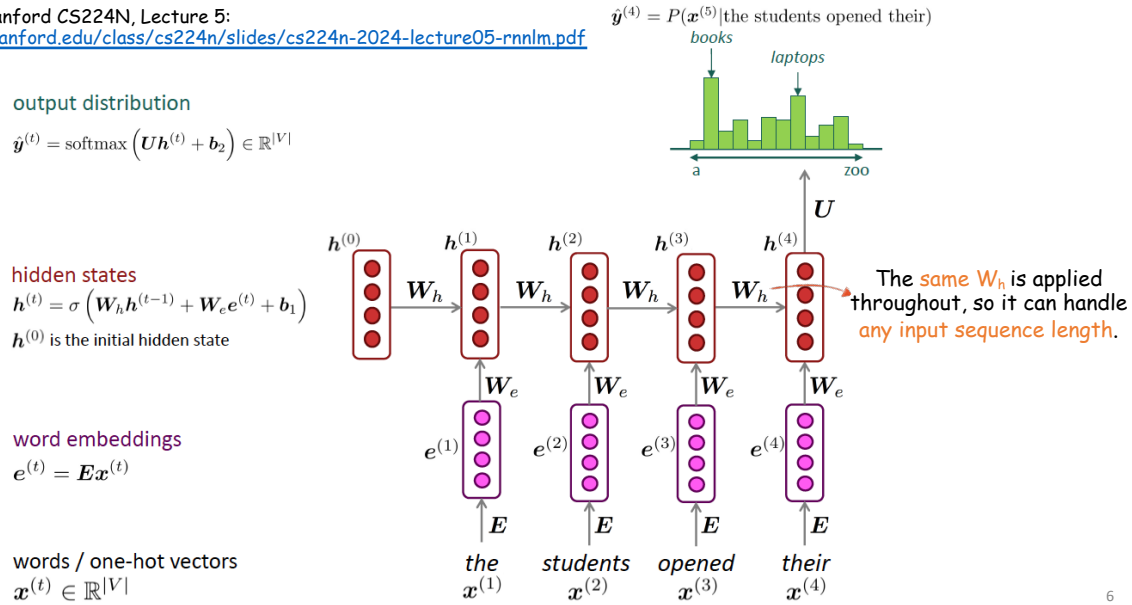
- N-gram models that "work" in the era of LLM: https://arxiv.org/abs/2401.17377

5

5

Recurrent Neural Network (RNN)

Reference: Stanford CS224N, Lecture 5: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture05-rnnlm.pdf



6

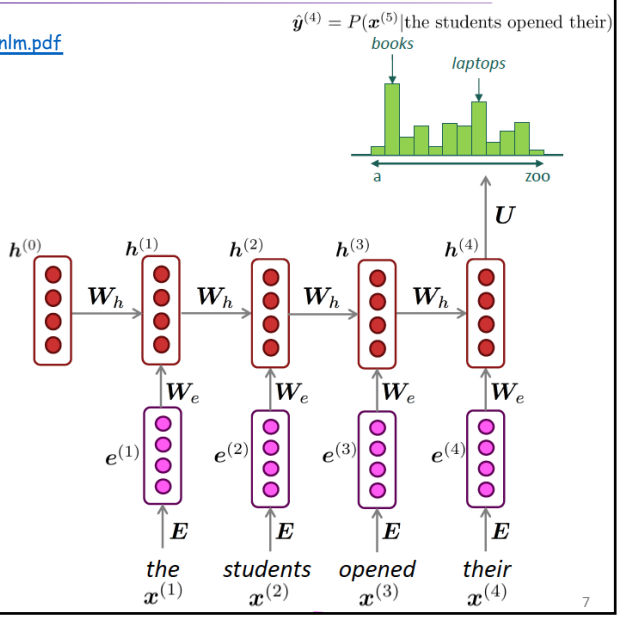
6

Recurrent Neural Network (RNN)

Reference: Stanford CS224N, Lecture 5:
https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture05-rnnlm.pdf

RNN Advantages:
- Can process any input sequence length.
- Computations (in theory) use information from many steps back.
- Same weights are applied to every step, so there's time-symmetry/invariance in how inputs are processed.

RNN Disadvantages:
- Recurrent computations are slow.
- In practice, it is challenging to access information from many steps back.



7

7

Training RNN

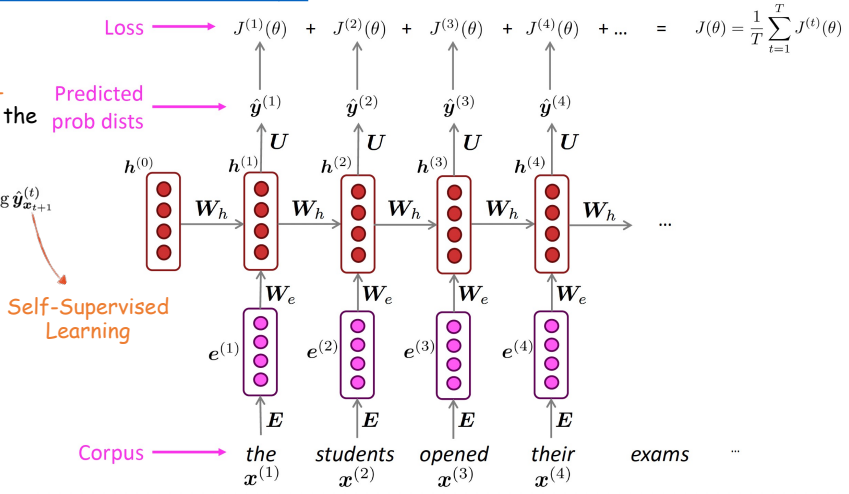
Reference: Stanford CS224N, Lecture 5:
https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture05-rnnlm.pdf

- Loss functions in step t is the cross-entropy between the true 1-hot and the predicted distribution:

$$J^{(t)}(\theta) = CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = -\sum_{w \in V} \boldsymbol{y}_w^{(t)} \log \hat{\boldsymbol{y}}_w^{(t)} = -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

- So, the overall loss for the entire training corpus is:

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} J^{(t)}(\theta) = \frac{1}{T}\sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$



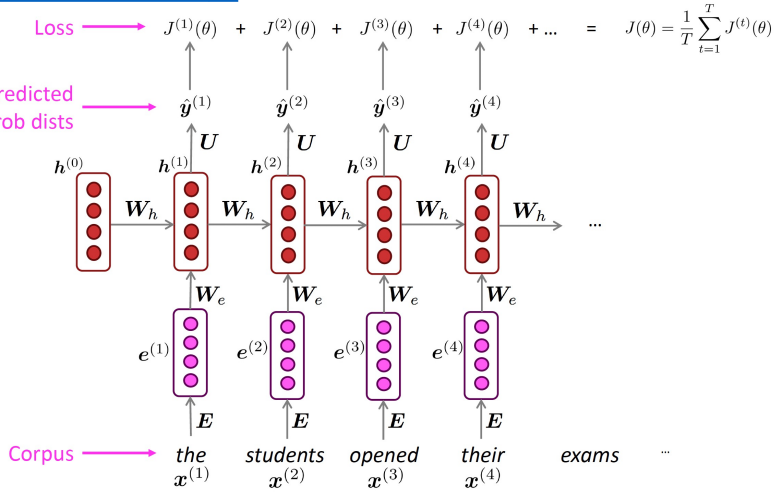
8

8

Training RNN

Reference: Stanford CS224N, Lecture 5:
https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture05-rnnlm.pdf

- Computing the loss and the gradients across the entire corpus is computationally too expensive.
- In practice, we leverage the idea of SGD: Compute loss and gradients, and update weights with batches of words/sentences. Then repeat on a new batch of sentences.



Vanishing (and Exploding) Gradient in RNN

Reference: Stanford CS224N, Lecture 5:
https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture05-rnnlm.pdf

Backpropagation through time

$$\frac{\partial L}{\partial W_h} \propto \sum_{1 \le k \le t} \left(\prod_{t \ge i > k} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial W_h}$$

Contribution of hidden state *k*

Length of the product proportional to how far *k* is from *t*

$$\frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{\partial h_{t-2}}{\partial h_{t-3}} \frac{\partial h_{t-3}}{\partial h_{t-4}} \frac{\partial h_{t-4}}{\partial h_{t-5}} \frac{\partial h_{t-5}}{\partial h_{t-6}} \frac{\partial h_{t-6}}{\partial h_{t-7}} \frac{\partial h_{t-7}}{\partial h_{t-8}} \frac{\partial h_{t-8}}{\partial h_{t-9}} \frac{\partial h_{t-9}}{\partial h_{t-10}} \frac{\partial h_{t-10}}{\partial W_h}$$

Contribution of hidden state *t-10*

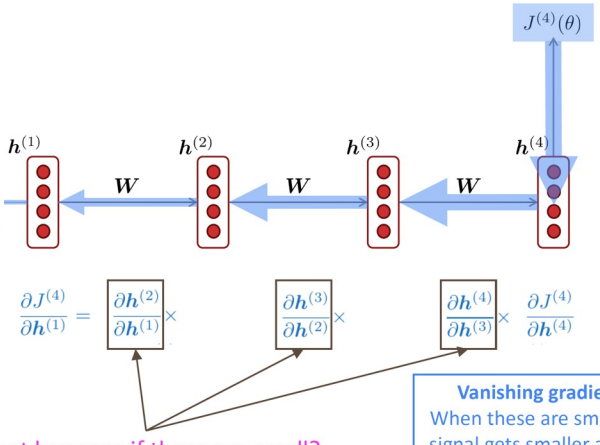
Vanishing vs. Exploding Gradient

On the difficulty of training recurrent neural networks
R Pascanu, T Mikolov, Y Bengio
International conference on machine learning, 2013 · proceedings.mlr.press

Abstract
There are two widely known issues with properly training recurrent neural networks, the vanishing and the exploding gradient problems detailed in Bengio et al.(1994). In this paper we attempt to improve the understanding of the underlying issues by exploring these problems from an analytical, a geometric and a dynamical systems perspective. Our analysis is used to justify a simple yet effective solution. We propose a gradient norm clipping strategy to deal with exploding gradients and a soft constraint for the vanishing

SHOW MORE

☆ Save 99 Cite Cited by 6901 Related articles All 11 versions »»



What happens if these are small?

Gradient signals from far away will be lost!
The weights W_h only capture near effects.

Vanishing gradient problem:
When these are small, the gradient signal gets smaller and smaller as it backpropagates further

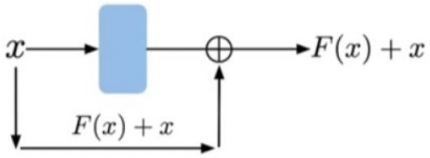
Gradient Clipping and Skip Connection

Reference: Stanford CS224N, Lecture 5:
https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture05-rnnlm.pdf

- The exploding gradient problem is relatively easier to address: Gradient Clipping.
- Intuition: Take a smaller step in the same direction.
- One idea to address vanishing gradient is to create direct and linear pass-through connections in the model: Residual/skip connections, attention, etc.

Algorithm 1 Pseudo-code for norm clipping

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if



Deep residual learning for image recognition
K He, X Zhang, S Ren, J Sun - … and pattern recognition, 2016 - openaccess.thecvf.com
… Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. …
★ 保存 99 引用 被引用次数：196717 相关文章 所有 76 个版本 »

11

11

Agenda

- Vanilla Recurrent Neural Nets (RNN)
- Long Short-Term Memory (LSTM)
- Sequence-to-sequence (Seq2seq)

12

12

Long Short-Term Memory RNN (LSTM)

Reference: Stanford CS224N, Lecture 6: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture06-fancy-rnn.pdf

- LSTM is a very commonly used RNN architecture that solves the vanishing gradient problem by disregarding the irrelevant past information based on the current information.
- LSTM was the dominant approach for most NLP tasks in 2013-2015.
- Very interesting history:
  - Everyone cites Hochreiter and Schmidhuber (1997), but the crucial part of modern LSTM comes from Gers et al. (2000).
  - Recognized as promising only after Graves et al. (2006) which invented CTC (connectionist temporal classification) for speech recognition.
  - Became well-known after Hinton brought LSTM to Google in 2013 (Graves was a student of Schmidhuber and a post-doc of Hinton).

Long short-term memory
S Hochreiter, J Schmidhuber - Neural computation, **1997** - ieeexplore.ieee.org
… (**short-term memory**, as opposed to **long**-term **memory** … learning what to put in **shortterm memory**, however, take too … and corresponding teacher signals are **long**. Although theoretically …
☆ Save 99 Cite Cited by 98734 Related articles All 45 versions »

Learning to forget: Continual prediction with LSTM
FA Gers, J Schmidhuber, F Cummins - Neural computation, 2000 - ieeexplore.ieee.org
Long short-term memory (LSTM; Hochreiter & Schmidhuber, 1997) can solve numerous tasks not solvable by previous **learning** algorithms for recurrent neural networks (RNNs). We …
☆ Save 99 Cite Cited by 7858 Related articles All 27 versions Web of Science: 1965 »

Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks
A Graves, S Fernández, F Gomez, J Schmidhuber
Proceedings of the 23rd international conference on Machine learning, 2006 · dl.acm.org
Many real-world sequence learning tasks require the prediction of sequences of labels from noisy, unsegmented input data. In speech recognition, for example, an acoustic signal is transcribed into words or sub-word units. Recurrent neural networks (RNNs) are powerful sequence learners that would seem well suited to such tasks. However, because they require pre-segmented training data, and post-processing to transform their outputs into label sequences, their applicability has so far been limited. This paper presents a
SHOW MORE
☆ Save 99 Cite Cited by 6329 Related articles All 26 versions »

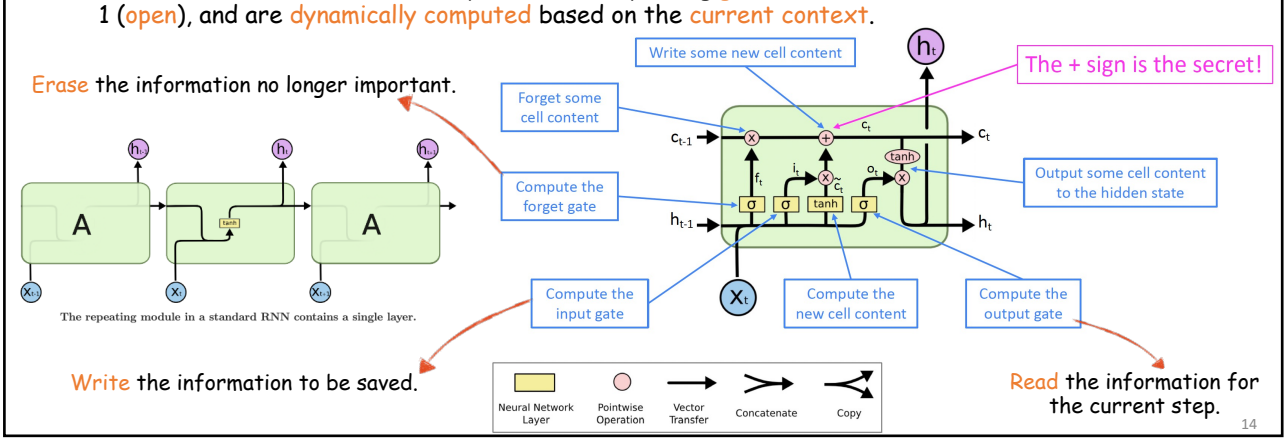
13

13

LSTM Model Details

Reference: Stanford CS224N, Lecture 6:
https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture06-fancy-rnn.pdf
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

- LSTM has a hidden state $h^{(t)}$ and a cell state $c^{(t)}$, which stores long-term information.
  - The LSTM model can read, erase, and write information from the cell state, much like RAM.
  - Read/erase/write is controlled by three corresponding gates, which take values between 0 (closed) and 1 (open), and are dynamically computed based on the current context.



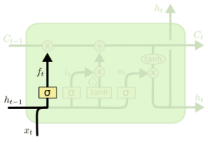
14

14

LSTM Model Details

Reference: Stanford CS224N, Lecture 6:
https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture06-fancy-rnn.pdf
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

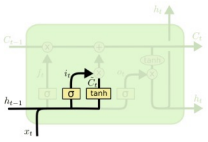
- Step 1: Decide how to forget the prior information.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Forget Gate

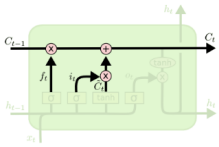
- Step 2: Decide the information needed to be stored to the cell.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

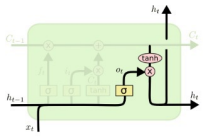
Input Gate

- Step 3: Update the cell information.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- Step 4: Decide the output.



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Output Gate

15

15

LSTM for Sentiment Classification

	text	stars	sentiment
1411425	i went here and ordered the barbiq burger and i got the meat in medium well i had to admit it was really good although the fries got in the way it was salty and i had to dip the fries with mayo and ketchup all and all i would give the fries 35 out of 5 the service of the burger was right on time i like the look of the restaurant with the fun and relaxed look of the burger joint worth the visit especially if you just want to try a good burger	3	neg
971153	after calling 3 difference companies number one were the only one that had the least wait time for our broken ac in the middle of summer we called on monday morning and they were able to send a technician out that night at 8 pm when the technician came he determined it was our blower motor that was the problem and told us they will have to order the parts for it it will take couple days the repair will probably get done on wednesday and they will call us on tuesday to let us know the time they told us the repair will be between 11 am 2 pm on wednesday the repair guy pulled up in uhaul and after going up to the attic to check out the blower told us he needed to go get more parts for it and will be back in the afternoon\n\ncome 5 pm still no sign of the guy so we called them back and they had to call us back and track down the repair guy the receptionist called back and said he will be back soon come 7 pm still no sign so i called again and they told me they are on their way the guys did not show up until 845 pm took them about 30 minutes for the repair \n\nin the end yes my ac was repaired in a timely manner i understand summer is the busiest time for them my issue is the lack of communication we were waiting and waiting and not knowing whats going on and had to keep calling them back also we never got a receipt for the repair which they said they would email us	3	neg
1270461	went here for dinner and left very full and satisfied the family that runs the restaurant is originally from globe so the mexican food here is similar to what youd find in the globemiami area stepping into this restaurant it feels similar to how other restaurants such as serranos and rositas feels comfy atmosphere serving homestyle no frills cuisine i had the special tonight 2 chicken enchiladas with green sauce with rice and beans 850 while the enchiladas looked a bit mangled since chicken pieces were sticking out of the tortilla and the tortilla itself looked prodded and broken the enchiladas themselves were quite tasty with the green sauce and the rice and beans im usually a lightweight when it comes to finishing meals but in this case i cleaned my plate the salsa tastes good but is very watery which makes it hard to eat with chips we ordered iced teas and they were refilled promptly as needed id definitely be interested in going here again for some tasty and filling meals	4	pos

```
data['sentiment'] = ['pos' if (x>3) else 'neg' for x in data['stars']]
data['text'] = data['text'].apply((lambda x: re.sub('[^a-zA-z0-9\s]','',x)))
for idx,row in data.iterrows():
    row[0] = row[0].replace('rt',' ')
data['text'] = [x.encode('ascii') for x in data['text']]

tokenizer = Tokenizer(nb_words=2500, lower=True,split=' ')
tokenizer.fit_on_texts(data['text'].values)
#print(tokenizer.word_index)  # To see the dicstionary
X = tokenizer.texts_to_sequences(data['text'].values)
X = pad_sequences(X)
```

```
embed_dim = 128
lstm_out = 200
batch_size = 32

model = Sequential()
model.add(Embedding(2500, embed_dim,input_length = X.shape[1], dropout = 0.2))
model.add(LSTM(lstm_out, dropout_U = 0.2, dropout_W = 0.2))
model.add(Dense(2,activation='softmax'))
model.compile(loss = 'categorical_crossentropy', optimizer='adam',metrics = ['accuracy'])
print(model.summary())
```

16

16

Evaluating Language Models

Reference: Stanford CS224N, Lecture 6:
https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture06-fancy-rnn.pdf
https://engineering.fb.com/2016/10/25/ml-applications/building-an-efficient-neural-language-model-over-a-billion-words/

- Perplexity: The standard metric for evaluating a language model.

$$\text{perplexity} = \prod_{t=1}^{T} \left(\frac{1}{P_{\text{LM}}(\boldsymbol{x}^{(t+1)} \mid \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})} \right)^{1/T}$$

Normalized by number of words

Inverse probability of corpus, according to Language Model

- Perplexity is the exponential of the cross-entropy loss, so the lower the better:

$$\text{Perplexity} = \prod_{t=1}^{T} \left(\frac{1}{\hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}} \right)^{1/T} = \exp\left(\frac{1}{T} \sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)} \right) = \exp(J(\theta))$$

RNNs greatly improve perplexity over n-grams.

Model	Perplexity
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	67.6
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3
RNN-2048 + BlackOut sampling (Ji et al., 2015)	68.3
Sparse Non-negative Matrix factorization (Shazeer et al., 2015)	52.9
LSTM-2048 (Jozefowicz et al., 2016)	43.7
2-layer LSTM-8192 (Jozefowicz et al., 2016)	30
Ours small (LSTM-2048)	43.9
Ours large (2-layer LSTM-2048)	39.8

5-Gram

DNN-based Methods
(RNN, LSTM, etc.)

Table 2. Comparison on 1B word in perplexity (lower the better). Note that Jozefowicz et al., uses 32 GPUs for training. We only use 1 GPU.

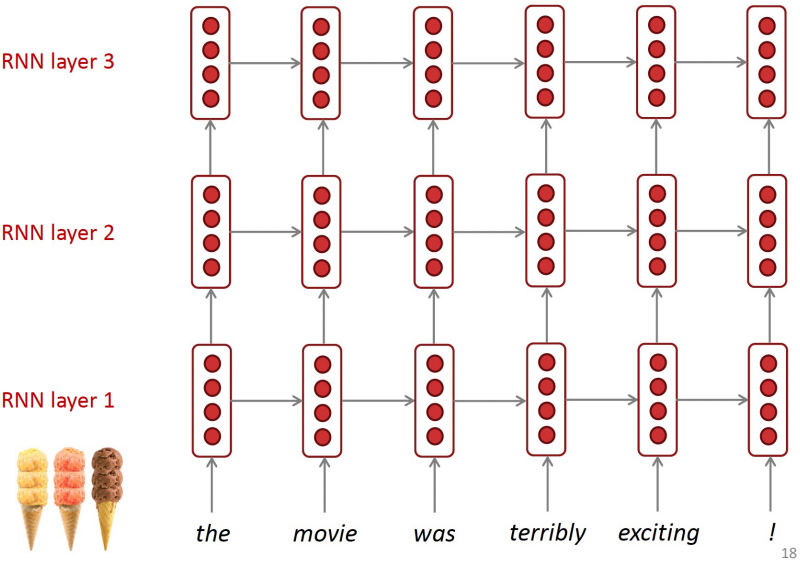
17

17

Multi-layer (Stacked) RNN

Reference: Stanford CS224N, Lecture 6: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture06-fancy-rnn.pdf

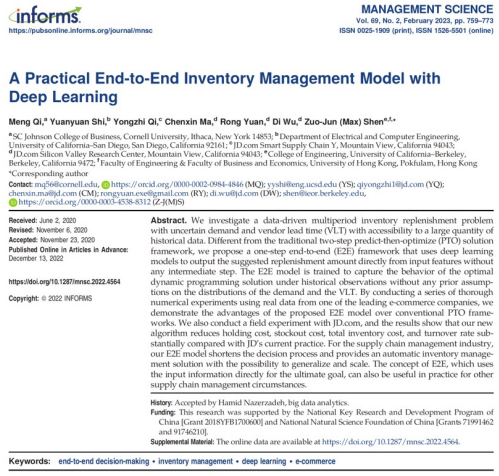
- RNNs are already deep on time dimension.
- We can also deepen them in another dimension by applying multiple RNNs, allowing for more complex representations and achieving better performances.
- In neural machine translation, 2-4 layers is best for encoder RNNs; 4 layers is best for decoder RNNs.
- 2 layers is much better than 1, but 3 may be a little better than 2.
- Transformer-based nets are usually much deeper.



18

18

Application of RNN: Inventory Management



informs
https://pubsonline.informs.org/journal/mnsc

MANAGEMENT SCIENCE
Vol. 69, No. 2, February 2023, pp. 759–773
ISSN 0025-1909 (print), ISSN 1526-5501 (online)

A Practical End-to-End Inventory Management Model with Deep Learning

Meng Qi,[a] Yuanyuan Shi,[b] Yongzhi Qi,[c] Chenxin Ma,[d] Rong Yuan,[d] Di Wu,[d] Zuo-Jun (Max) Shen[e,f,*]

[a] SC Johnson College of Business, Cornell University, Ithaca, New York 14853; [b] Department of Electrical and Computer Engineering, University of California–San Diego, San Diego, California 92161; [c] JD.com Smart Supply Chain Y, Mountain View, California 94043; [d] JD.com Silicon Valley Research Center, Mountain View, California 94043; [e] College of Engineering, University of California–Berkeley, Berkeley, California 9472; [f] Faculty of Engineering & Faculty of Business and Economics, University of Hong Kong, Pokfulam, Hong Kong
*Corresponding author
Contact: mq56@cornell.edu, https://orcid.org/0000-0002-0984-4846 (MQ); yyshi@eng.ucsd.edu (YS); qiyongzhi1@jd.com (YQ); chenxin.ma@jd.com (CM); rongyuan.exe@gmail.com (RY); di.wu@jd.com (DW); shen@ieor.berkeley.edu, https://orcid.org/0000-0003-4538-8312 (Z-JMS)

Received: June 2, 2020
Revised: November 6, 2020
Accepted: November 23, 2020
Published Online in Articles in Advance: December 13, 2022
https://doi.org/10.1287/mnsc.2022.4564

Abstract. We investigate a data-driven multiperiod inventory replenishment problem with uncertain demand and vendor lead time (VLT) with accessibility to a large quantity of historical data. Different from the traditional two-step predict-then-optimize (PTO) solution framework, we propose a one-step end-to-end (E2E) framework that uses deep learning models to output the suggested replenishment amount directly from input features without any intermediate step. The E2E model is trained to capture the behavior of the optimal dynamic programming solution under historical observations without any prior assumptions on the distributions of the demand and the VLT. By conducting a series of thorough numerical experiments using real data from one of the leading e-commerce companies, we demonstrate the advantages of the proposed E2E model over conventional PTO frameworks. We also conduct a field experiment with JD.com, and the results show that our new algorithm reduces holding cost, stockout cost, total inventory cost, and turnover rate substantially compared with JD's current practice. For the supply chain management industry, our E2E model shortens the decision process and provides an automatic inventory management solution with the possibility to generalize and scale. The concept of E2E, which uses the input information directly for the ultimate goal, can also be useful in practice for other supply chain management circumstances.

History: Accepted by Hamid Nazerzadeh, big data analytics.
Funding: This research was supported by the National Key Research and Development Program of China [Grant 2018YFB1700600] and National Natural Science Foundation of China [Grants 71991462 and 91746210].
Supplemental Material: The online data are available at https://doi.org/10.1287/mnsc.2022.4564.

Keywords: end-to-end decision-making • inventory management • deep learning • e-commerce

- RNN is not that frequently used in business research, because it does NOT directly produce text representations.
- Use multi-quantile RNNs to provide end-to-end predictions from features to the optimal inventory decisions, whereas most of the literature applies the predict-then-optimize paradigm.
- A field experiment shows that the e2e approach substantially reduces the inventory costs compared with some naïve benchmarks.

A practical end-to-end inventory management model with deep learning
M Qi, Y Shi, Y Qi, C Ma, R Yuan, D Wu, ZJ Shen
Management Science, 2023 · pubsonline.informs.org

We investigate a data-driven multiperiod inventory replenishment problem with uncertain demand and vendor lead time (VLT) with accessibility to a large quantity of historical data. Different from the traditional two-step predict-then-optimize (PTO) solution framework, we propose a one-step end-to-end (E2E) framework that uses deep learning models to output the suggested replenishment amount directly from input features without any intermediate step. The E2E model is trained to capture the behavior of the optimal dynamic

SHOW MORE

☆ Save 99 Cite Cited by 65 Related articles All 4 versions Web of Science: 5 »

19

19

Application of RNN: Detecting FTD

Psychiatry Research 304 (2021) 114135

Contents lists available at ScienceDirect

Psychiatry Research

journal homepage: www.elsevier.com/locate/psychres

Detecting formal thought disorder by deep contextualized word representations

Justyna Sarzynska-Wawer [*,a], Aleksander Wawer [1,b], Aleksandra Pawlak [2,c], Julia Szymanowska [2,c], Izabela Stefaniak [d], Michal Jarkiewicz [3,d], Lukasz Okruszek [a]

[a] Institute of Psychology, Polish Academy of Sciences, Jaracza 1, 00-378 Warszawa, Poland
[b] Institute of Computer Science, Polish Academy of Sciences, Jana Kazimierza 5, 01-248 Warszawa, Poland
[c] University of Social Sciences and Humanities, Chodakowska 19/31, 03-815 Warszawa, Poland
[d] Institute of Psychiatry and Neurology, Sobieskiego 9, 02-957 Warszawa, Poland

ARTICLE INFO

Keywords:
Schizophrenia
Language
Natural language processing
Deep learning

ABSTRACT

Computational linguistics has enabled the introduction of objective tools that measure some of the symptoms of schizophrenia, including the coherence of speech associated with formal thought disorder (FTD). Our goal was to investigate whether neural network based utterance embeddings are more accurate in detecting FTD than models based on individual indicators. The present research used a comprehensive Embeddings from Language Models (ELMo) approach to represent interviews with patients suffering from schizophrenia (N=35) and with healthy people (N=35). We compared its results to the approach described by Bedi et al. (2015), referred to here as the coherence model. Evaluations were also performed by a clinician using the Scale for the Assessment of Thought, Language and Communication (TLC). Using all six TLC questions the ELMo obtained an accuracy of 80% in distinguishing patients from healthy people. Previously used coherence models were less accurate at 70%. The classifying clinician was accurate 74% of the time. Our analysis shows that both ELMo and TLC are sensitive to the symptoms of disorganization in patients. In this study methods using text representations from language models were more accurate than those based solely on the assessment of FTD, and can be used as measures of disordered language that complement human clinical ratings.

- How to detect formal thought disorder (FTD)?
- Embeddings from LSTM language models (ELMo) can more accurately detect/predict FTD than individual indicators (benchmark: coherence model, which can somehow be viewed as traditional NLP method).
- Accuracy (N=70, 35 healthy and 35 patients):
  - ELMo: 80%
  - Coherence models: <70%
  - Clinician: 74%

[HTML] Detecting formal thought disorder by deep contextualized word representations
J Sarzynska-Wawer, A Wawer, A Pawlak… - Psychiatry …, 2021 - Elsevier

Computational linguistics has enabled the introduction of objective tools that measure some of the symptoms of schizophrenia, including the coherence of speech associated with formal thought disorder (FTD). Our goal was to investigate whether neural network based utterance embeddings are more accurate in detecting FTD than models based on individual indicators. The present research used a comprehensive Embeddings from Language Models (ELMo) approach to represent interviews with patients suffering from schizophrenia …

☆ Save 99 Cite Cited by 14561 Related articles All 24 versions Web of Science: 74 »

20

20

Agenda

- Vanilla Recurrent Neural Nets (RNN)
- Long Short-Term Memory (LSTM)
- Sequence-to-sequence (Seq2seq)

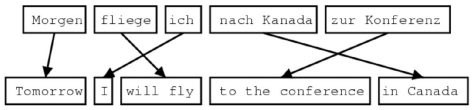
21

21

Neural Machine Translation (NMT)

Reference: Stanford CS224N, Lecture 6: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture06-fancy-rnn.pdf

- NMT is a way to do machine translation with a single end-to-end neural network: Sequence-to-sequence (seq2seq), which involves 2 RNNs.
- Machine translation is highly nontrivial and once was a huge research field in CS and NLP.



1519年600名西班牙人在墨西哥登陆，去征服几百万人口的阿兹特克帝国，初次交锋他们损兵三分之二。

In 1519, six hundred Spaniards landed in Mexico to conquer the Aztec Empire with a population of a few million. They lost two thirds of their soldiers in the first clash.

translate.google.com (2009): 1519 600 Spaniards landed in Mexico, millions of people to conquer the Aztec empire, the first two-thirds of soldiers against their loss.

translate.google.com (2013): 1519 600 Spaniards landed in Mexico to conquer the Aztec empire, hundreds of millions of people, the initial confrontation loss of soldiers two-thirds.

translate.google.com (2015): 1519 600 Spaniards landed in Mexico, millions of people to conquer the Aztec empire, the first two-thirds of the loss of soldiers they clash.

22

22

Seq2Seq for NMT

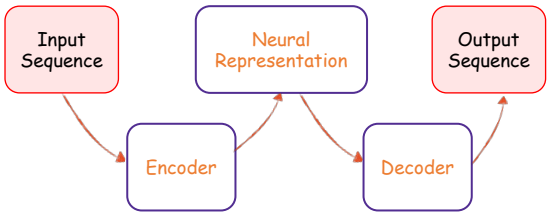
Reference: Stanford CS224N, Lecture 6: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture06-fancy-rnn.pdf

- Seq2seq is a Conditional Language Model:
  - Predicting the next word of the target sentence y conditioned on the source sentence x and prior texts.

$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots\underbrace{P(y_T|y_1,\ldots,y_{T-1},x)}$$

Probability of next target word, given target words so far and source sentence *x*

- Encoder-decoder architecture: Encoder takes input and produces a neural representation; Decoder produces output based on that neural representation.
  - Seq2seq: both input and output are sequences.
  - Summarization: Long text → short text
  - Dialogue: previous utterances → next utterance
  - Parsing: Input text → output parse as a sequence
  - Code generation → Natural language → Python code

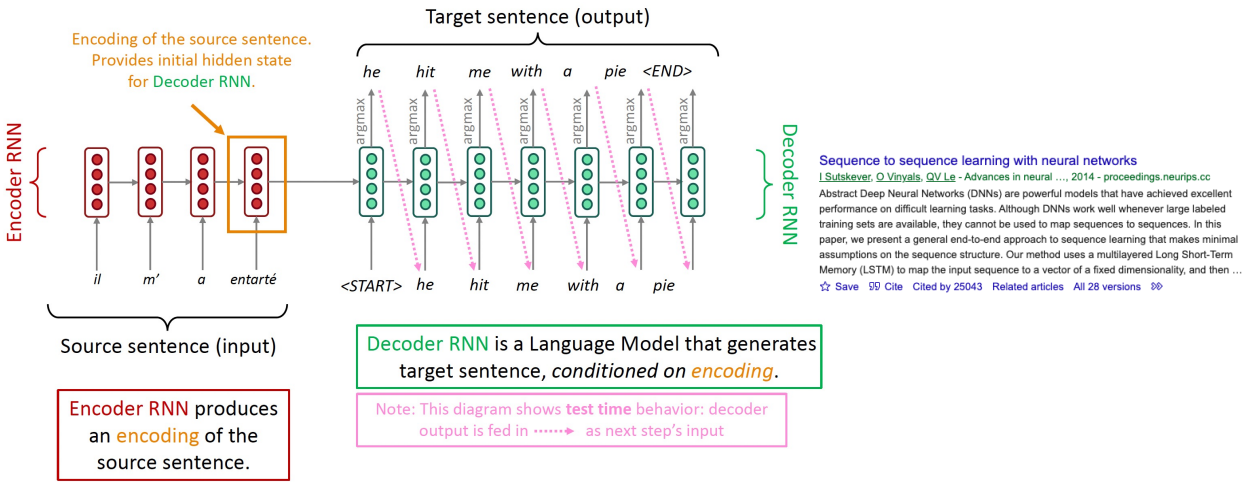


23

23

Seq2Seq Architecture

Reference: Stanford CS224N, Lecture 6: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture06-fancy-rnn.pdf



Encoder RNN produces an encoding of the source sentence.

Decoder RNN is a Language Model that generates target sentence, *conditioned on encoding*.

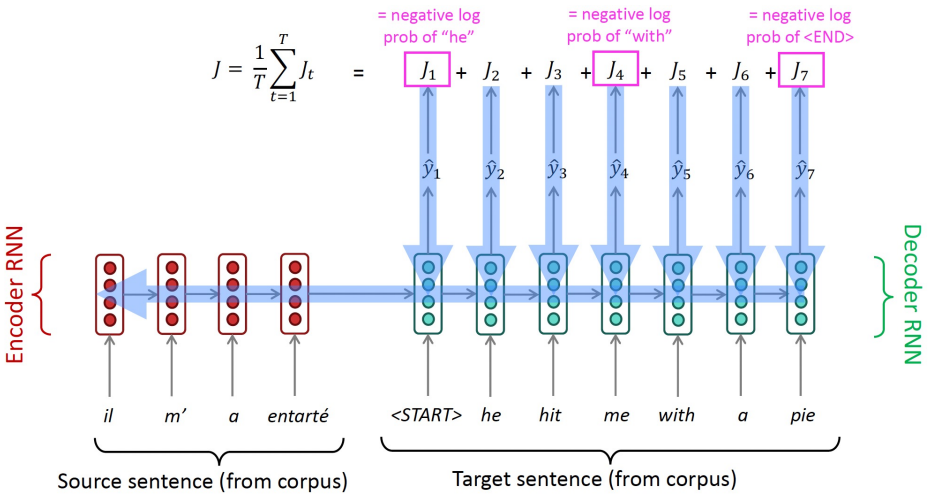
Note: This diagram shows **test time** behavior: decoder output is fed in ⋯→ as next step's input

24

24

Seq2Seq Training

Reference: Stanford CS224N, Lecture 6: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture06-fancy-rnn.pdf



Seq2seq is optimized as a **single system.** Backpropagation operates "*end-to-end*".

25

25

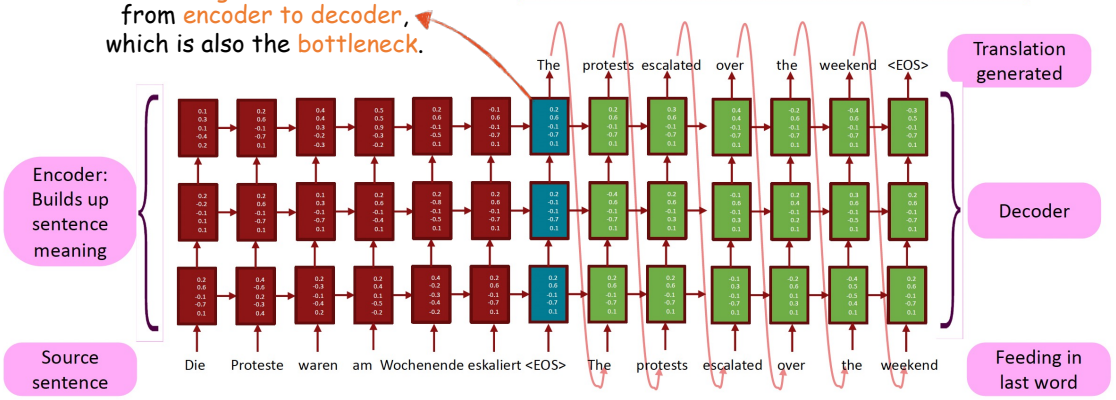
Multi-Layer Seq2Seq

Reference: Stanford CS224N, Lecture 6: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture06-fancy-rnn.pdf

[Sutskever et al. 2014; Luong et al. 2015]

The hidden states from RNN layer *i* are the inputs to RNN layer *i*+1

Conditioning: Information flow from encoder to decoder, which is also the bottleneck.



What's next: Attention Mechanism and Transformer!

26

26