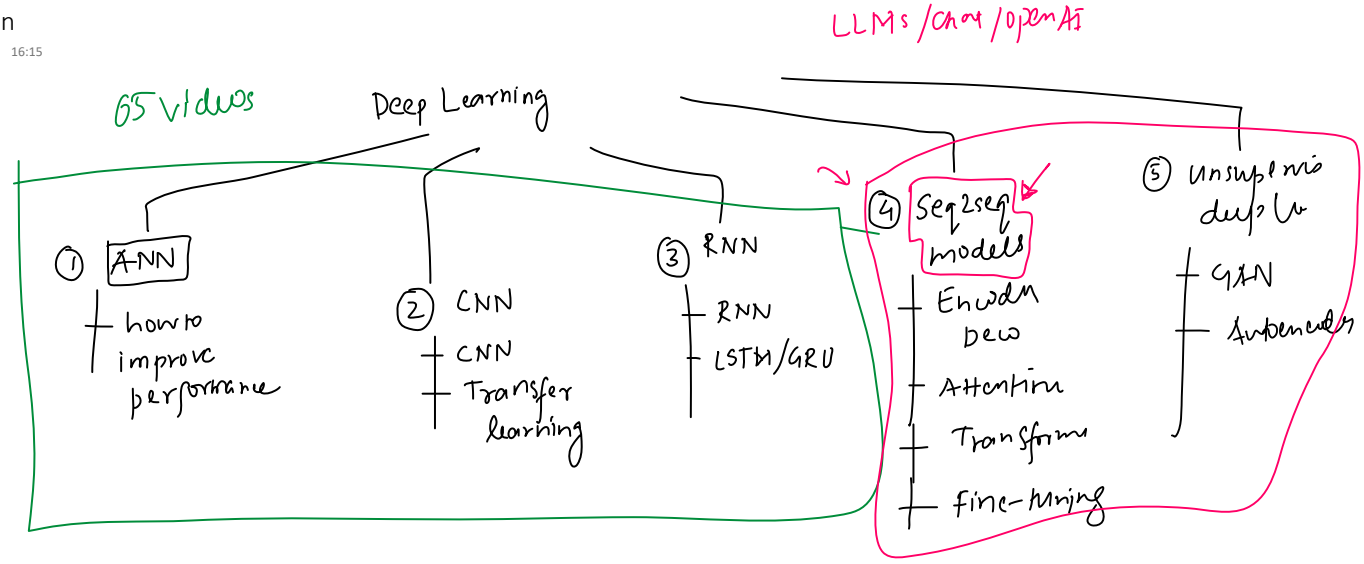
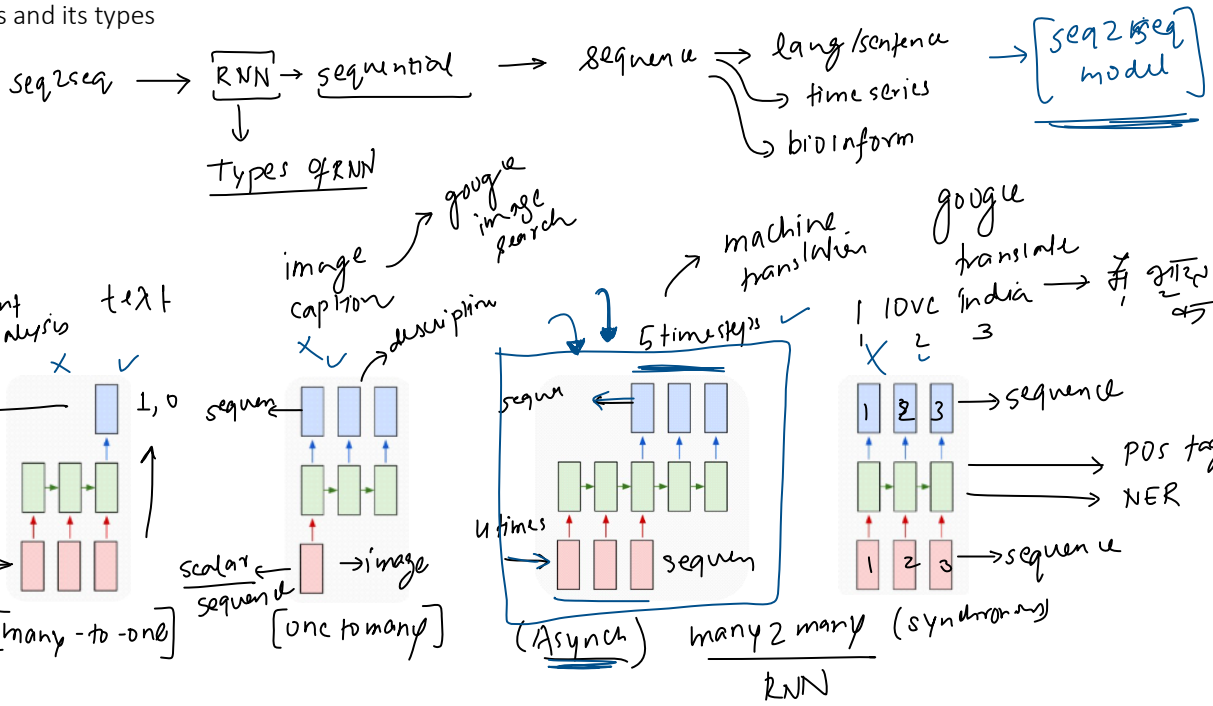
Introduction

16 November 2023 16:15



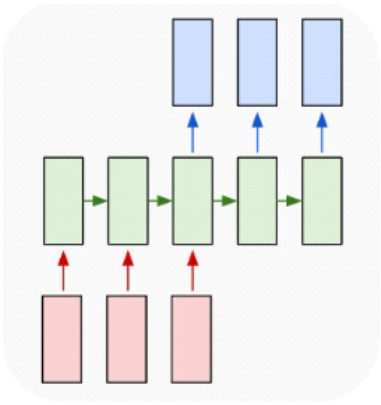
Sequence Tasks and its types

16 November 2023 16:16



Seq2Seq Tasks

16 November 2023 16:16



NLP

Seq2seq → machine trans

1) text summari → seq → seq

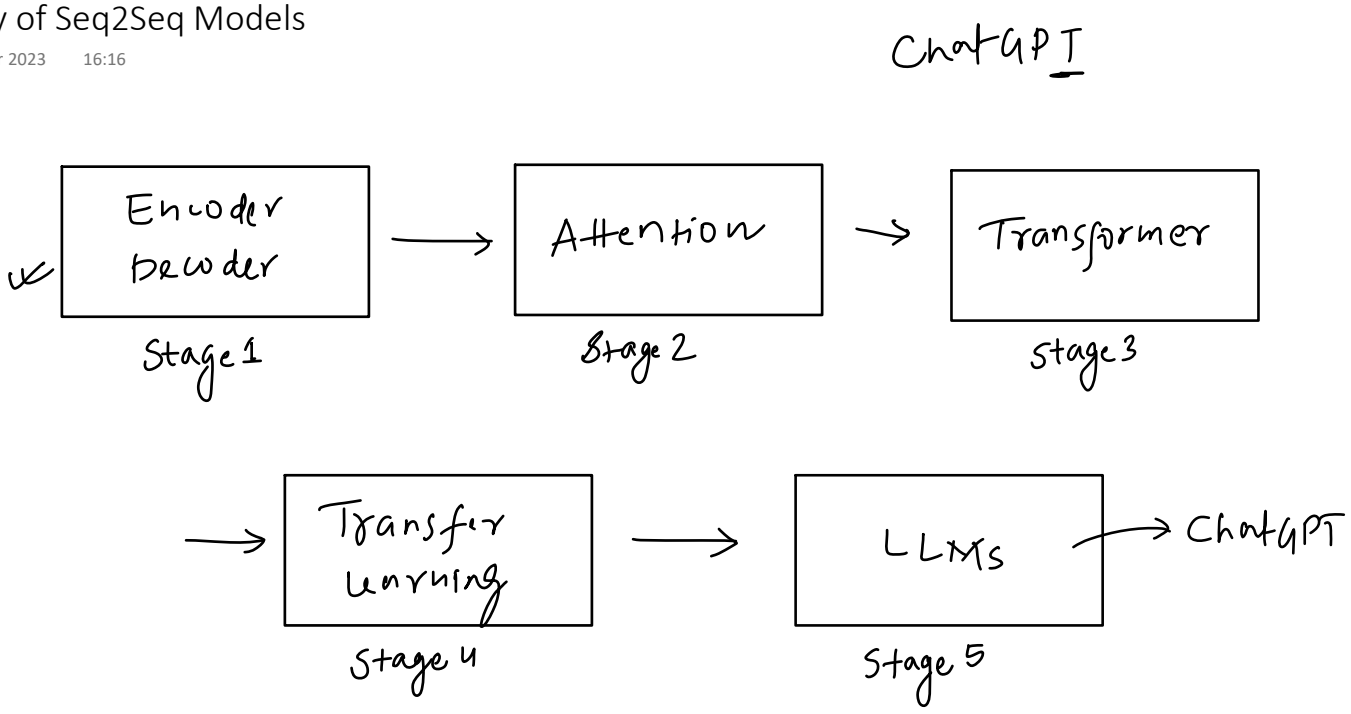
2) Question answer → seq → seq (knowledge base)

3) chatbot → input (text) → output (text)

4) speech-to-text → seq seq → seq2seq → chatgpt

History of Seq2Seq Models

16 November 2023 16:16



Stage 1 - Encoder Decoder Architecture

18 November 2023 16:16

2014 Seminal

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever
Google
ilyasu@google.com

Oriol Vinyals
Google
vinyals@google.com

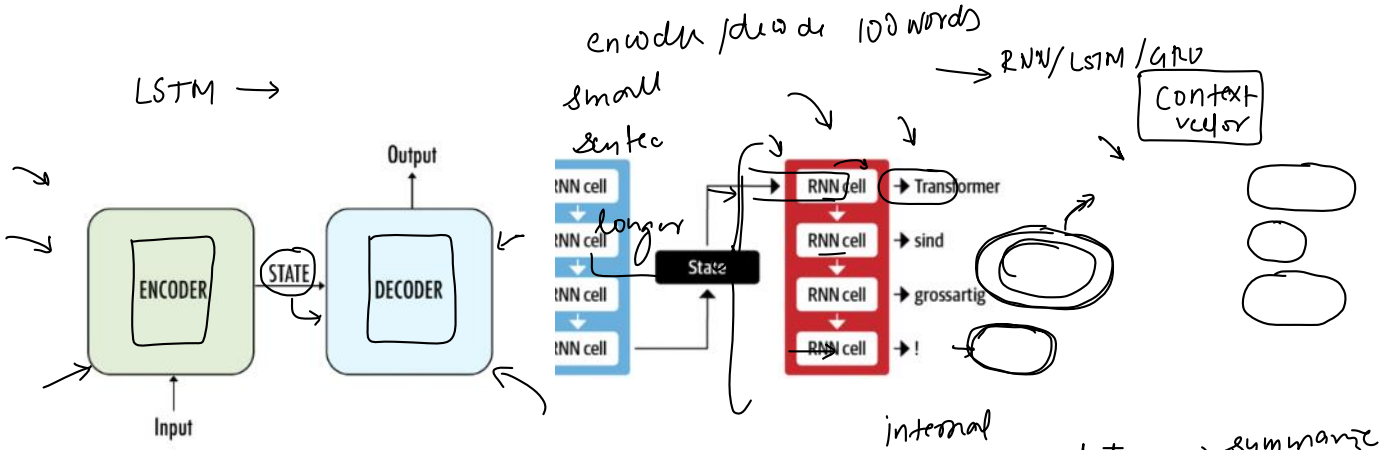
Quoc V. Le
Google
qvl@google.com

Abstract

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French translation task from the WMT'14 dataset, the translations produced by the LSTM achieve a BLEU score of 34.8 on the entire test set, where the LSTM's BLEU score was penalized on out-of-vocabulary words. Additionally, the LSTM did not have difficulty on long sentences. For comparison, a phrase-based SMT system achieves a BLEU score of 33.3 on the same dataset. When we used the LSTM to rerank the 1000 hypotheses produced by the aforementioned SMT system, its BLEU score increases to 36.5, which is close to the previous best result on this task. The LSTM also learned sensible phrase and sentence representations that are sensitive to word order and are relatively invariant to the active and the passive voice. Finally, we found that reversing the order of the words in all source sentences (but not target sentences) improved the LSTM's performance markedly, because doing so introduced many short term dependencies between the source and the target sentence which made the optimization problem easier.

Ilya Sutskever → cofounder OpenAI

seq2seq
↓
diff
↳ encoder decoder



seq2seq → input → output
machine translation
eng → hindi

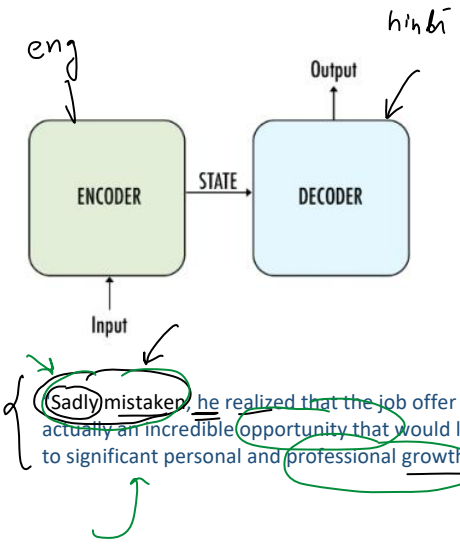
I love India → मैं भारत से प्यार करता हूँ

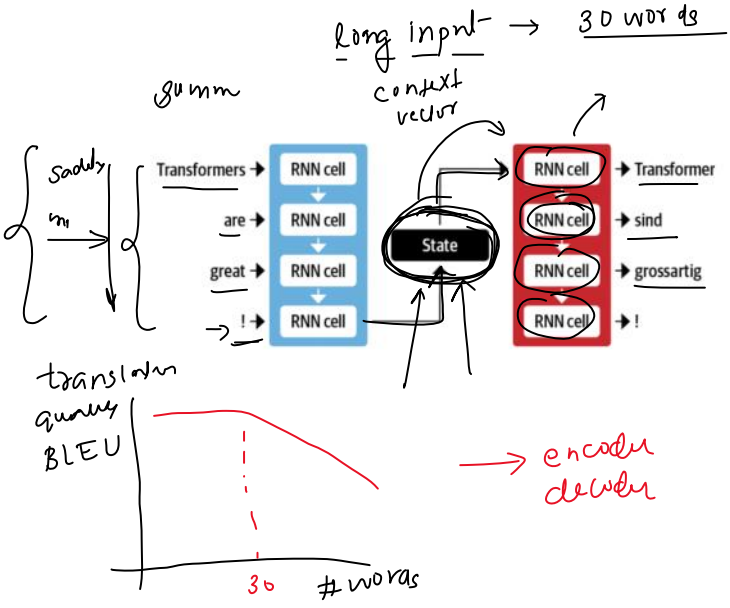
process word by word → compress

Attention mechanism

Stage 2 - Attention Mechanism

20 November 2023 10:59



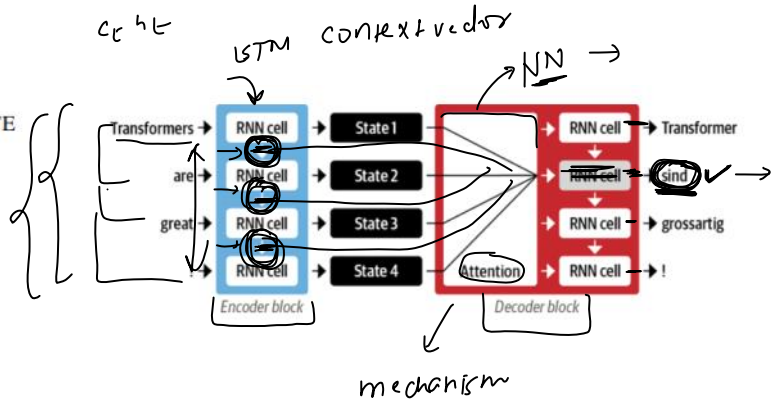


2015 Attention

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal



ABSTRACT

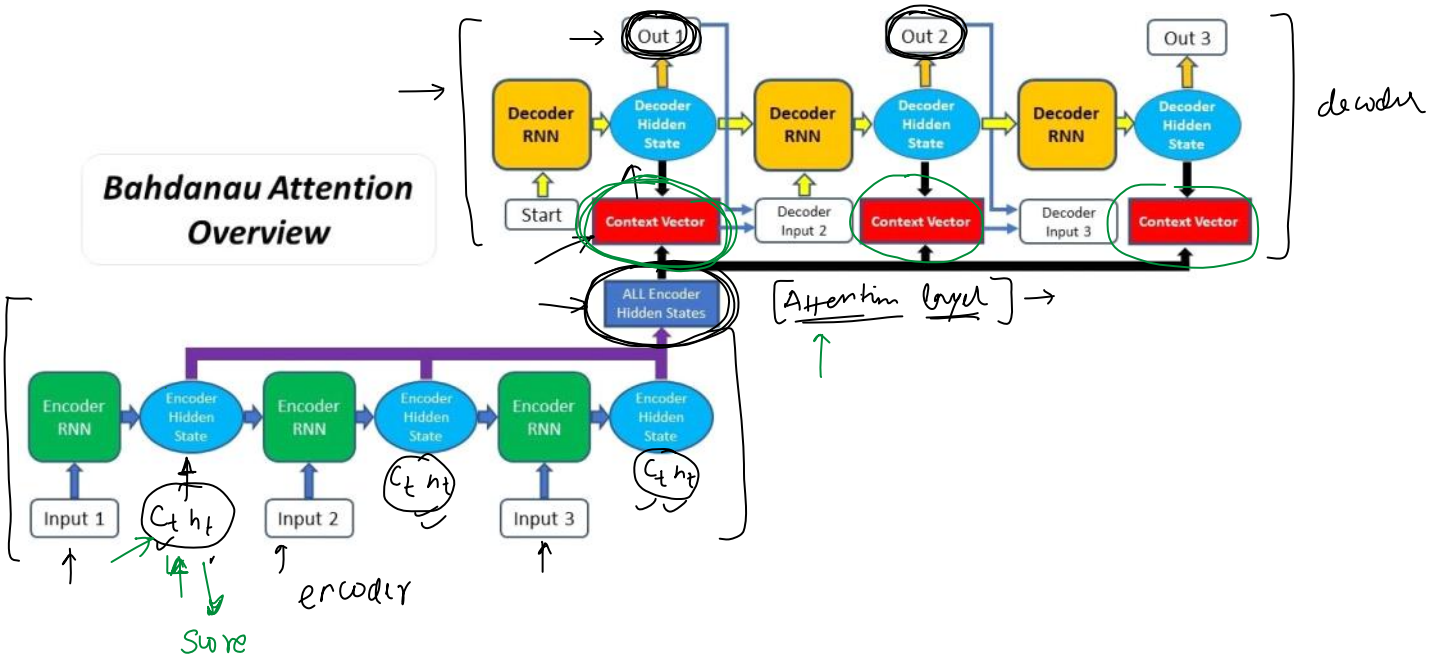
Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

1 INTRODUCTION

Neural machine translation is a newly emerging approach to machine translation, recently proposed by Kalchbrenner and Blunsom (2013), Sutskever *et al.* (2014) and Cho *et al.* (2014b). Unlike the traditional phrase-based translation system (see, e.g., Koehn *et al.*, 2003) which consists of many small sub-components that are tuned separately, neural machine translation attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation.

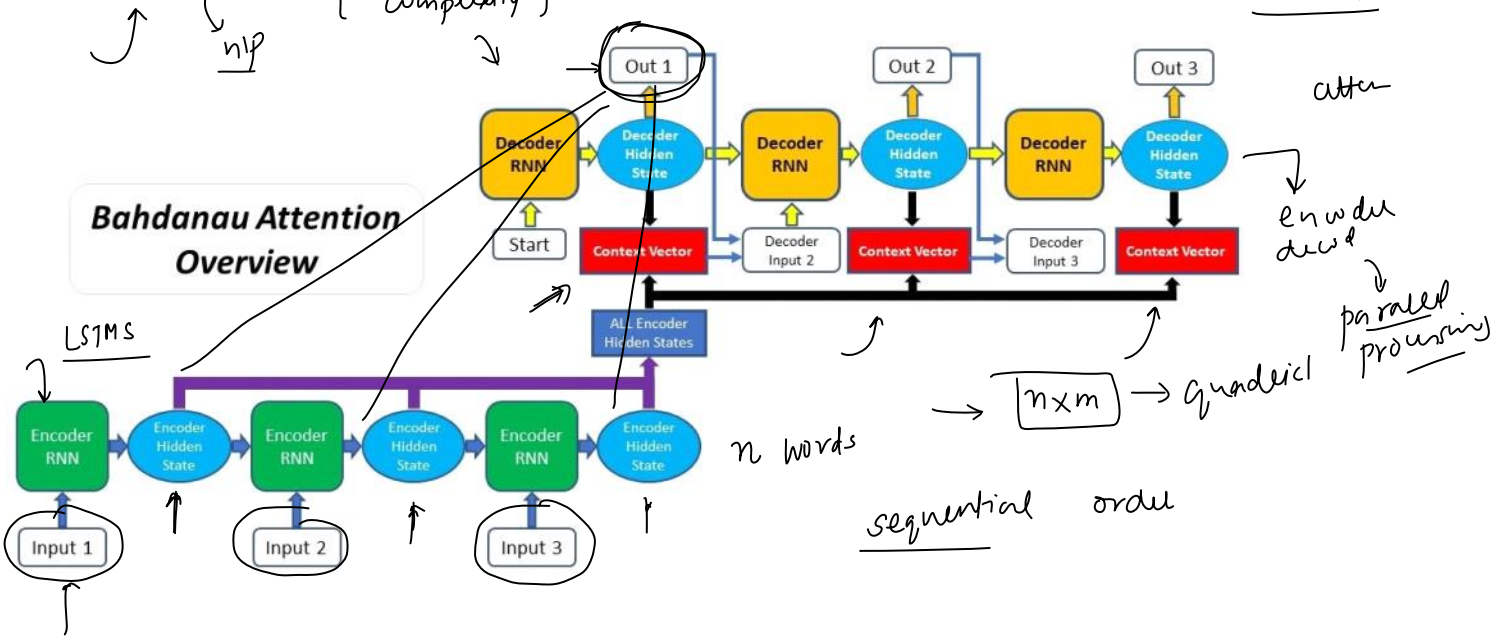
Most of the proposed neural machine translation models belong to a family of *encoder–decoders* (Sutskever *et al.*, 2014; Cho *et al.*, 2014a), with an encoder and a decoder for each language, or involve a language-specific encoder applied to each sentence whose outputs are then compared (Hermann and Blunsom, 2014). An encoder neural network reads and encodes a source sentence into a fixed-length vector. A decoder then outputs a translation from the encoded vector. The whole encoder–decoder system, which consists of the encoder and the decoder for a language pair, is jointly trained to maximize the probability of a correct translation given a source sentence.

A potential issue with this encoder–decoder approach is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences, especially those that are longer than the sentences in the training corpus. Cho *et al.* (2014b) showed that indeed the performance of a basic encoder–decoder deteriorates rapidly as the length of an input sentence increases.


Bahdanau Attention Overview
Out 1
Out 2
Out 3
Decoder RNN
Decoder Hidden State
Start
Context Vector
Decoder Input 2
Decoder Input 3
ALL Encoder Hidden States
Encoder RNN
Encoder Hidden State
Input 1
Input 2
Input 3
decoder
Attention layer
encoder
Score

Stage 3 - Transformers

20 November 2023 12:18



Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

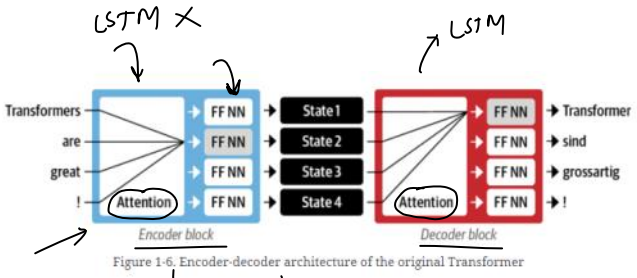


Figure 1-6. Encoder-decoder architecture of the original Transformer

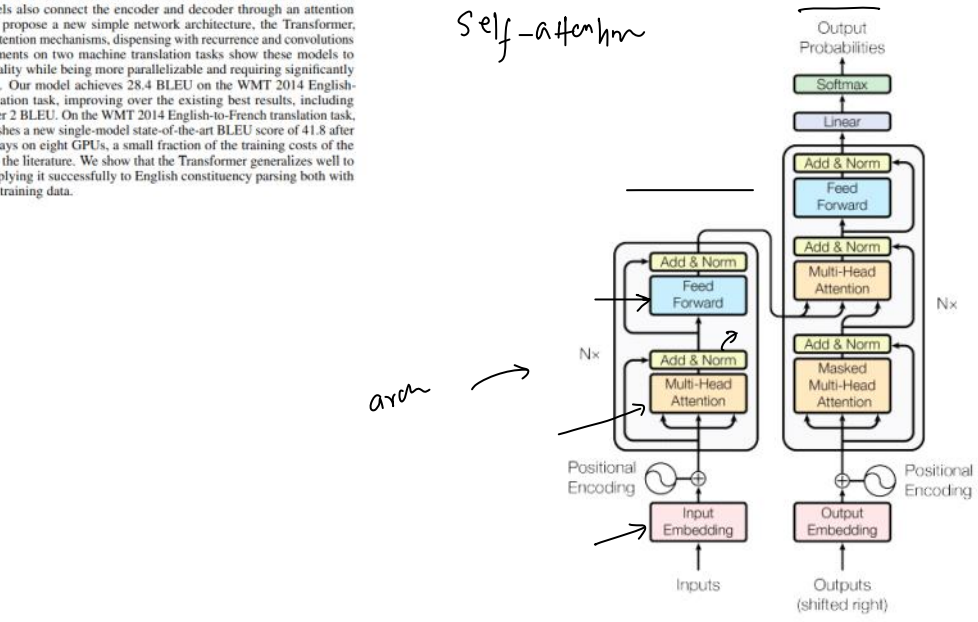
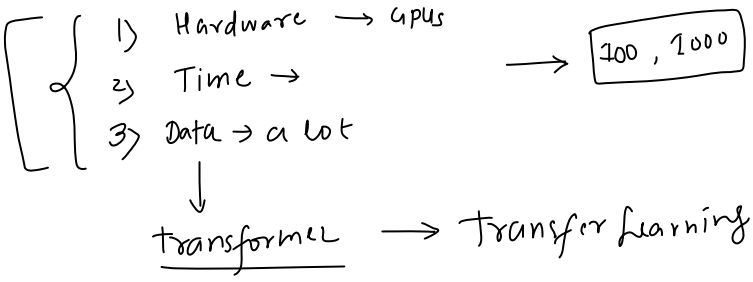


Figure 1: The Transformer - model architecture.

Stage 4 - Transfer Learning

20 November 2023 15:39



ULMFiT

Universal Language Model Fine-tuning for Text Classification

Jeremy Howard*
fast.ai
University of San Francisco
j@fast.ai

Sebastian Ruder*
Insight Centre, NUI Galway
Aylien Ltd., Dublin
sebastian@ruder.io

2018 → vision

transfer → NLP tasks

ULMFit

primary pretrain

Abstract

Inductive transfer learning has greatly impacted computer vision, but existing approaches in NLP still require task-specific modifications and training from scratch. We propose Universal Language Model Fine-tuning (ULMFiT), an effective transfer learning method that can be applied to any task in NLP, and introduce techniques that are key for fine-tuning a language model. Our method significantly outperforms the state-of-the-art on six text classification tasks, reducing the error by 18-24% on the majority of datasets. Furthermore, with only 100 labeled examples, it matches the performance of training from scratch on 100× more data. We open-source our pretrained models and code[1].

1 Introduction

Inductive transfer learning has had a large impact on computer vision (CV). Applied CV models (including object detection, classification, and segmentation) are rarely trained from scratch, but instead are fine-tuned from models that have been pretrained on ImageNet, MS-COCO, and other datasets (Sharif Razavian et al., 2014; Long et al., 2015a; He et al., 2016; Huang et al., 2017).

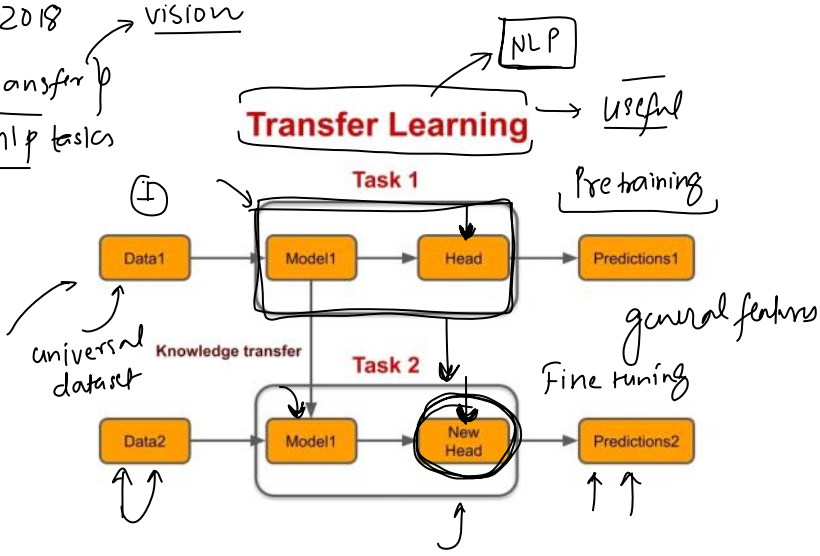
Text classification is a category of Natural Language Processing (NLP) tasks with real-world applications such as spam, fraud, and bot detection (Jindal and Liu, 2007; Ngai et al., 2011; Chu et al., 2012), emergency response (Caragea et al., 2011), and commercial document classification, such as for legal discovery (Roitblat et al., 2010).

While Deep Learning models have achieved state-of-the-art on many NLP tasks, these models are trained from scratch, requiring large datasets, and days to converge. Research in NLP focused mostly on *transductive* transfer (Blitzer et al., 2007). For *inductive* transfer, fine-tuning pretrained word embeddings (Mikolov et al., 2013), a simple transfer technique that only targets a model's first layer, has had a large impact in practice and is used in most state-of-the-art models. Recent approaches that concatenate embeddings derived from other tasks with the input at different layers (Peters et al., 2017; McCann et al., 2017; Peters et al., 2018) still train the main task model from scratch and treat pretrained embeddings as fixed parameters, limiting their usefulness.

In light of the benefits of pretraining (Erhan et al., 2010), we should be able to do better than *randomly initializing* the remaining parameters of our models. However, inductive transfer via fine-tuning has been unsuccessful for NLP (Mou et al., 2016). Dai and Le (2015) first proposed fine-tuning a language model (LM) but require millions of in-domain documents to achieve good performance, which severely limits its applicability.

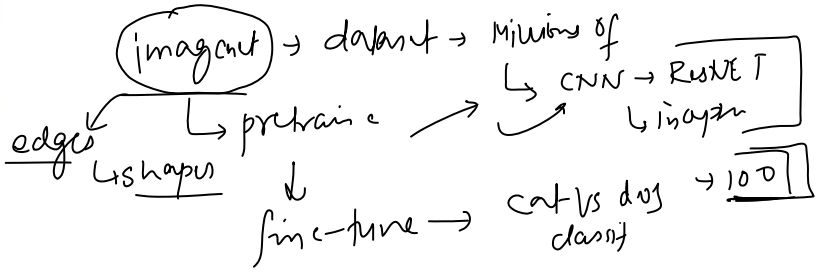
We show that not the idea of LM fine-tuning but our lack of knowledge of how to train them effectively has been hindering wider adoption. LMs overfit to small datasets and suffered catastrophic forgetting when fine-tuned with a classifier. Compared to CV, NLP models are typically more shallow and thus require different fine-tuning methods.

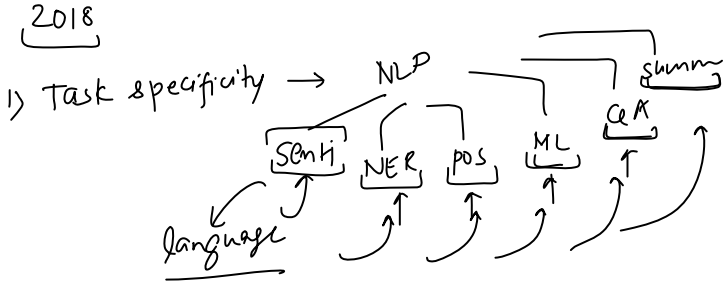
We propose a new method, Universal Language Model Fine-tuning (ULMFiT) that addresses these issues and enables robust inductive transfer learning for any NLP task, akin to fine-tuning ImageNet



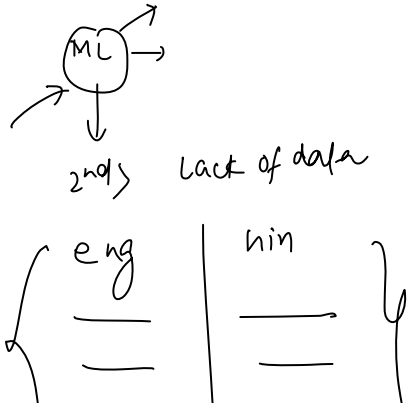
Transfer learning (TL) is a technique in which knowledge learned from a task is re-used in order to boost performance on a related task.

For example, for image classification, knowledge gained while learning to recognize cars could be applied when trying to recognize trucks.

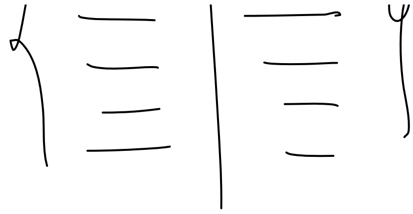




NLP → transfer learning



NLP → transfer learning



Pretraining → Machine transl X

↳ language modelling → pretraining → sucessful → (2)

NLP task → NLP/DL model next word predi

I live in india. and the capital is ___ New Delhi

Language modelling as a Pretraining task

↳ unsupervised pretrain task

1) Rich feature learning

The hotel was exceptionally clean, yet the service was ___ bad / pathetic

→ know trans

↓

text classi / quest. | text sum | NER / pos ...

mt (labe → supervised labeled

2) Huge avail of data

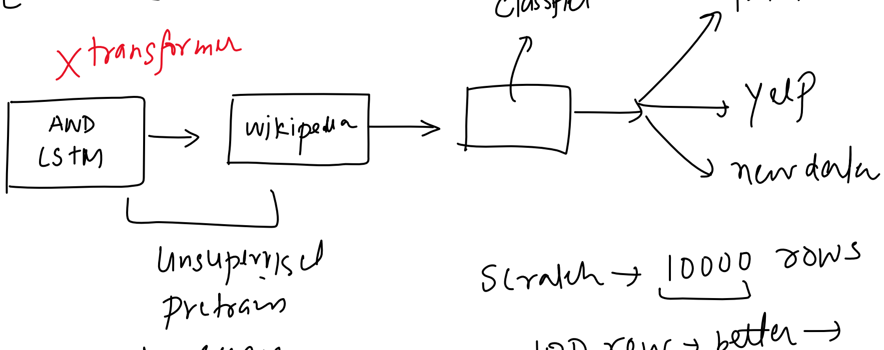
pdf → dataset labelling

eng | hin

→ unsupervised task

fine tuning

[ULMFiT]



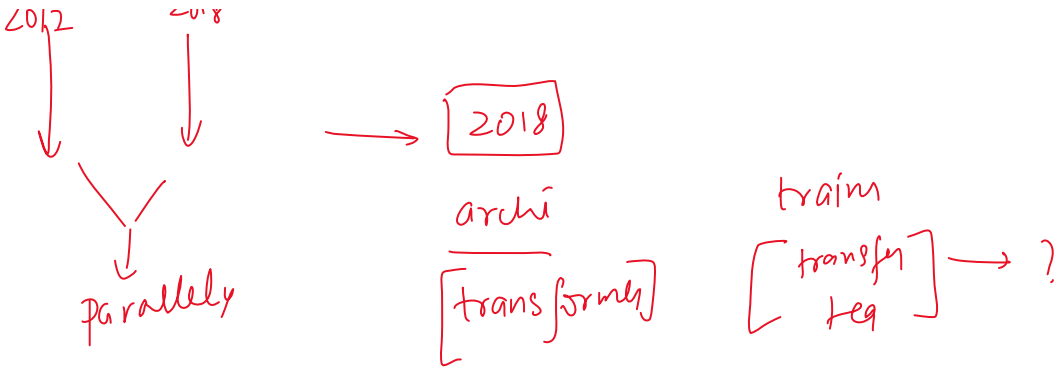
→ model ↓ test

Scratch → 10000 rows

100 row → better →

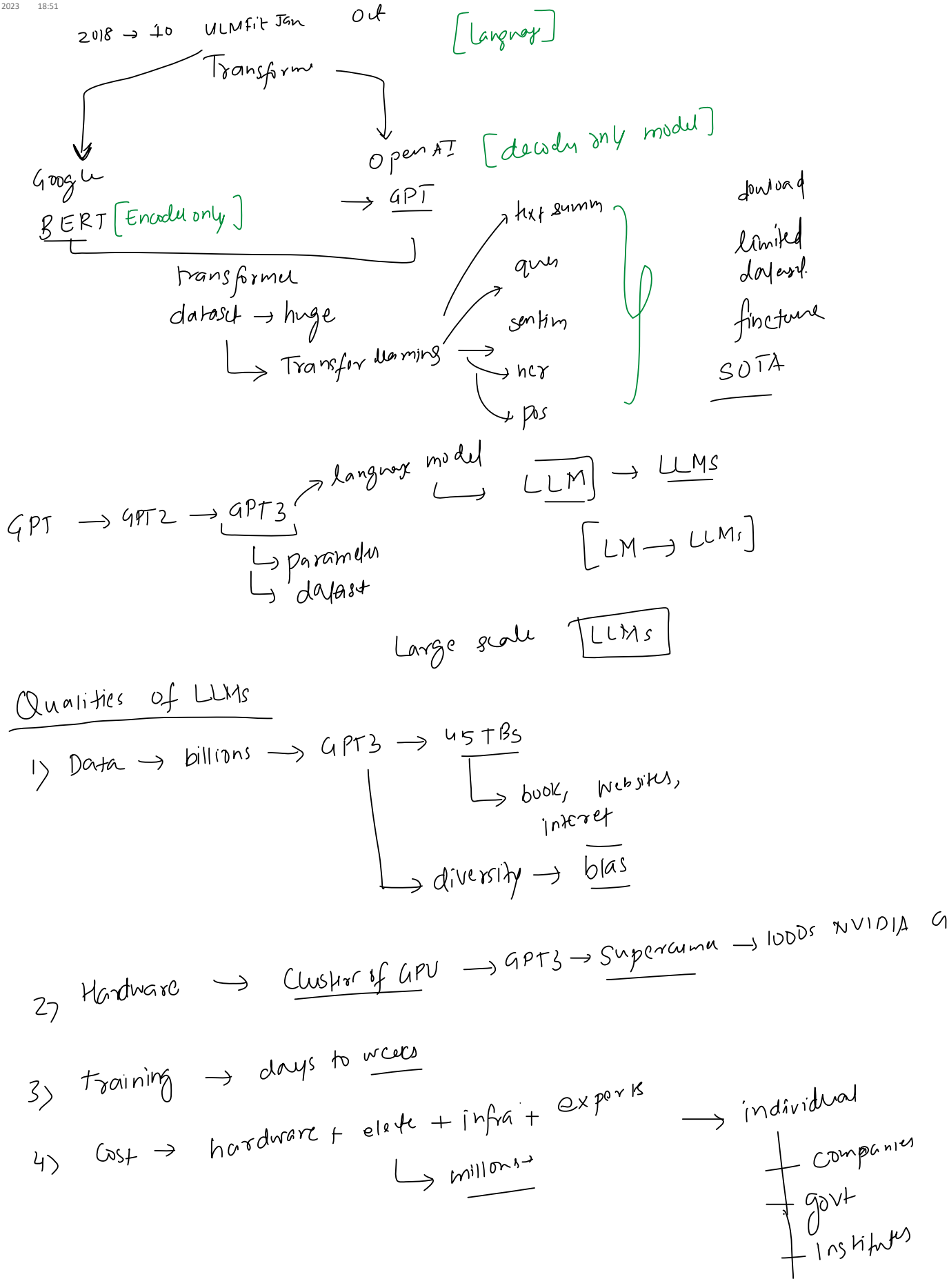
↓ State of the art

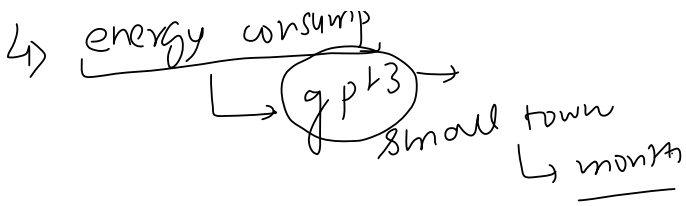
2012 2018


2012
2018
parallely
2018
archi
transformer
train
transfer
eq
?

Stage 5 - LLMs

20 November 2023 18:51




energy consump
gpt3
small town
month

The Grand Finale - ChatGPT

22 November 2023 10:13

GPT3 → chatGPT

[diff]

GPT → model

[ChatGPT] → application

Chat-bot

NOV

hp → chatgpt → Bard, Jasper

intel → GPT → Bard, Jasper

dell

ASW

GPT3 → [ChatGPT]

1) RLHF → Reinf lea from human feed

- 1 Supervised Fine tuning → dataset → labeled
- 2 Reinforce → prompt produces
  - responses
  - human → respone rank

2) Incorporate safety and ethical guideline

- minize biases

3) improvent in contextual point

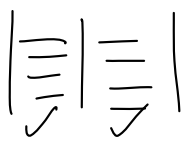
- context retain → maintain context → dialogu Conver

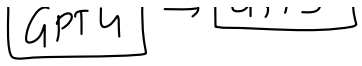
4) Dialogue specific training

- conversation
- better unders → dialogue lang → patterns

5) ChatGPT continous imp → human feedback usu

train → refining → GPT4 → GPT3


train → gpt


GPT4

Lecture: 68 (Encoder-Decoder)

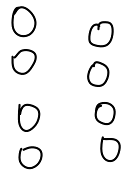
Seq2Seq Data

08 December 2023 17:12

cgpa | iq | placement

1) Tabular → [ANN] ×

DNN



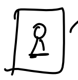
3) Sequential → textual
↘ timeseries

↓

_ _ _ _

↓ RNN
↙ LSTMs ↘ GRUs

2) Image data → [2d grid] → CNN ×



4) Seq2seq data — difficult

[input seq → output seq] — machine translation

nice to meet you
↓
आप से मिल कर अच्छा लगा

Challenges

input → sent → eng → variable length
output → sent → hindi → variable length → variable length
↓
[lstm / gru] → input
[output]

3 words eng → 3 words

i) input এ sentence থাকবে like English (variable length)

ii) output " " " " hindi (" ")

iii) input এ 3 টা word থাকলে output এ সেইটা নাও থাকতে পারে।

Variable length আমরা কিছুটা handle করতে শিখেছি। কিন্তু সেইটা input এ, GRU/LSTM ব্যবহার করে। Output এ কিভাবে handle করবে এটা এখন শিখতে হবে।

Sequence-sequence prob solve করার জন্য আমরা Encoder & Decoder Architecture আনবো।

Before Starting

08 December 2023 19:24

Prerequisite

- RNN / LSTM

Plan of attack

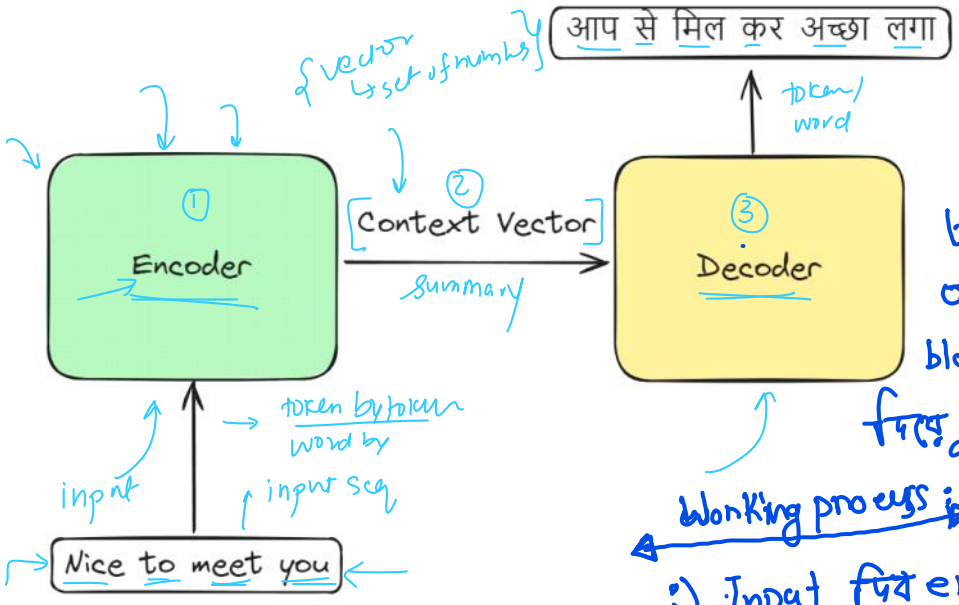
→ simple version
→ deep
→ improvements

High Level Overview

08 December 2023 17:14

Translation as an Example

Machine tran → exam



Encoder-Decoder এ দুইটা block থাকে। একটা Encoder আরেকটা Decoder। এই দুটা block এর context vector দিয়ে যুক্ত থাকে।

Working process :

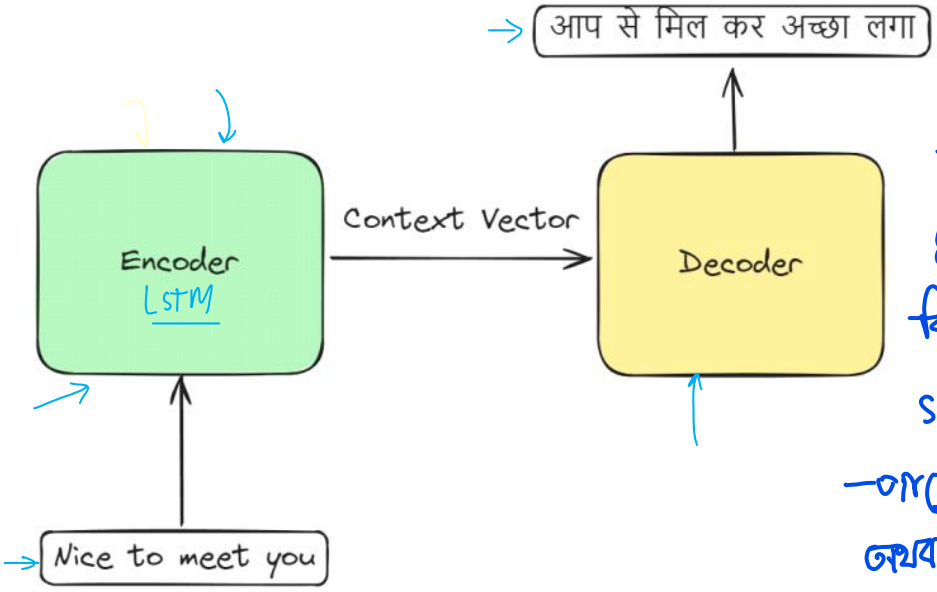
i) Input দিব encoder এ token by token or word by word.

ii) Encoder input vector গুলোকে যুক্ত করে summarize করবে। এর summarization output হবে একটা vector যেটাকে আমরা context vector বলেছি।

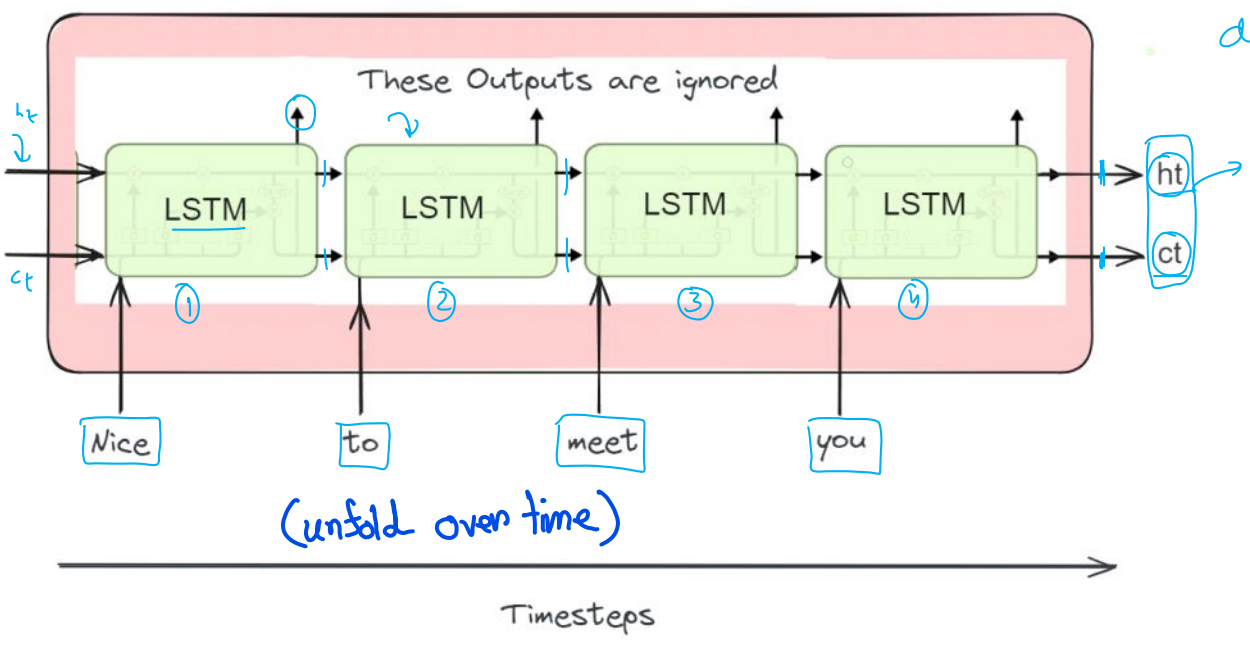
iii) Decoder এই context vector-কে যুক্ত করে word by word or token by token অন্য language এ translation দিবে।

What's under the hood?

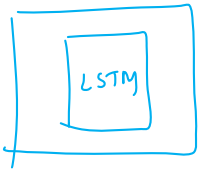
08 December 2023 17:14



অর্থাৎ, আমাদের endecoder & decoder এর মধ্যে যেমন কিছু থাকা দরকার যেইটা sequence কে process করতে পারে আর এটা হচ্ছে LSTM অথবা GRU। Research paper এ LSTM ব্যবহার করা হয়েছে।



decoder



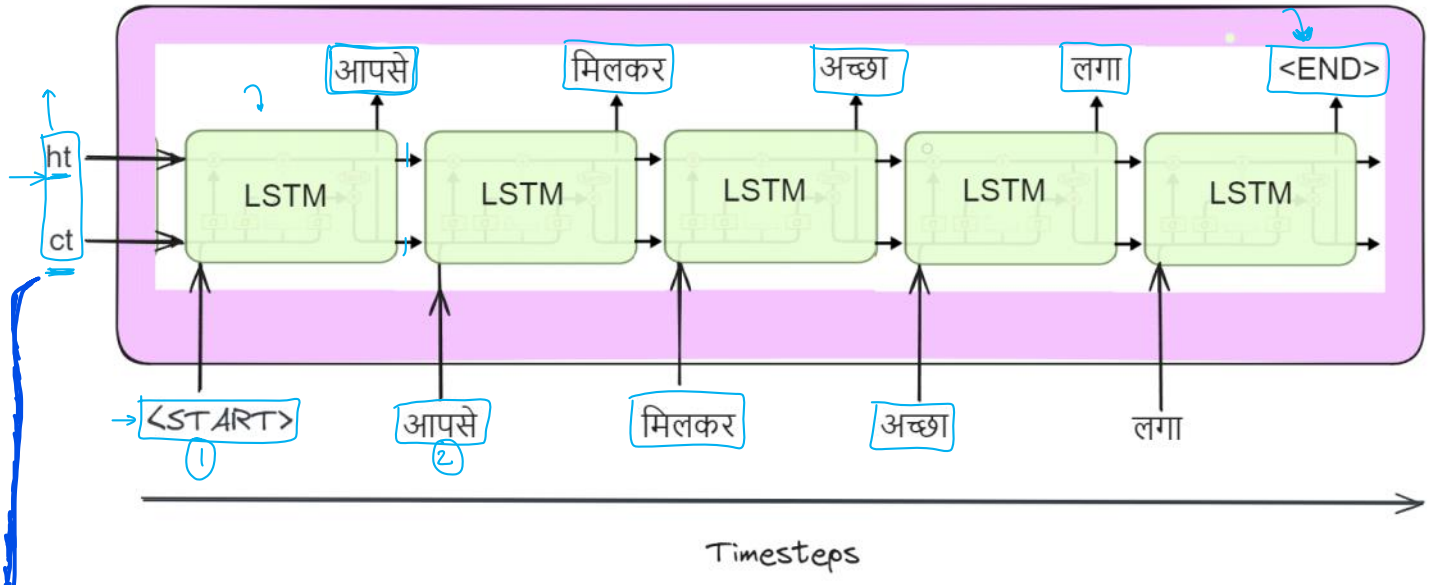
unfold over time

final representation

context vector

LSTM / GRU → RNN

* আমরা LSTM or GRU ব্যবহার করি। কিন্তু, RNN না ব্যবহার RNN এ vanishing Gradient প্রবলেম দেখা দেয়। *

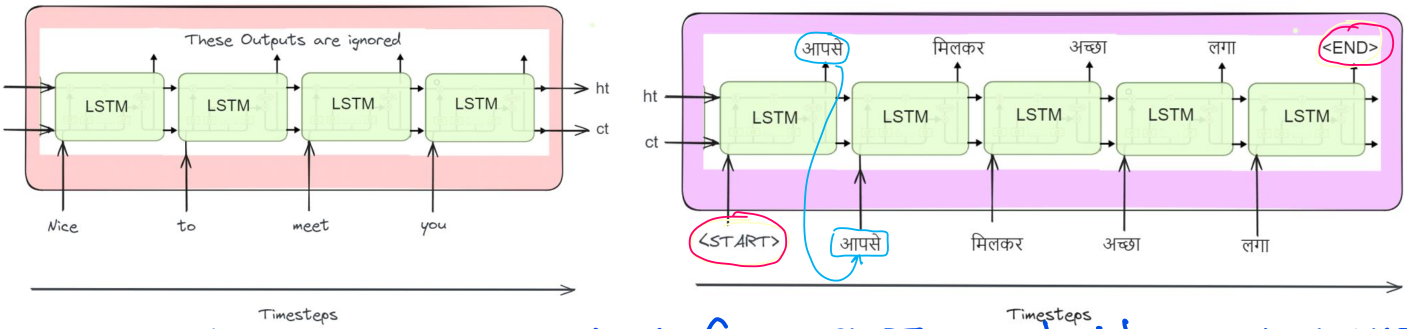


→ Encoder থেকে যেই output পাই, সেইটা decoder এ input হিসেবে যাই।

→ এখন আমরা দুইটা special symbol দেখ যেন encoder & decoder starting & ending বুঝতে পারে।

→ timestep-1 থেকে যেই output পাবো সেইটা timestep-1 এর জন্য input হিসেবে কাজ করে। এমন, চলতে থাকবে যতক্ষন না পর্যন্ত 2nd special symbol <end> পাচ্ছি।

encoder & decoder একই সাথে train হয়। তাই training প্রসেস বুঝার জন্য ২টা diagram একসাথে রাখবো।



Machine translation কে আমরা dataset হিসেবে ব্যবহার করে training process বুঝবো।

Data set for training:

Eng	Bangla
i) Think about it	এটা সম্পর্কে ভাবো
ii) come in	ভিতরে আসো

→ OHE (total unique 5 words)

eng: think [10000], about [01000]

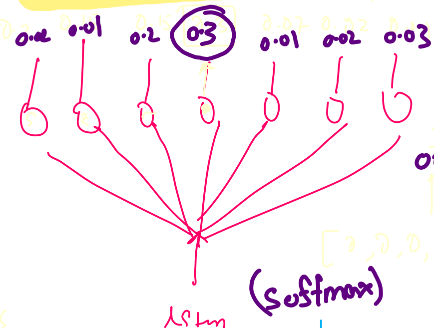
bangla: <start> <end> [1000000], <end> [0100000]

Tokenize the data set:

row1 — [Think, about, it] → [ভাবো, সম্পর্কে, এটা]

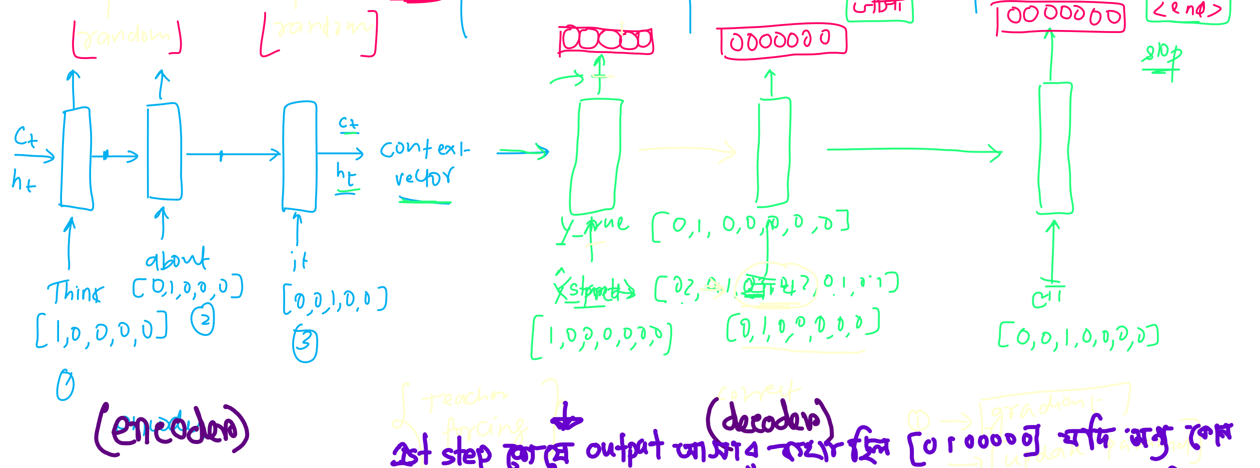
row2 — [come, in] → [আসো, ভিতরে]

[0 0 0 1 0 0 0] → means আসো



যার probability বেশি সেটাই হবে output।

Encoder এ আমরা প্রথম row input দিবো timestep এর ভিত্তিতে word by word encoder এ যাবে। Decoder এ প্রথমে special symbol start যাবে। Decoder এর output এ একটা softmax layer থাকে যেখানে input (বাংলাতে ৭)টা node থাকবে।

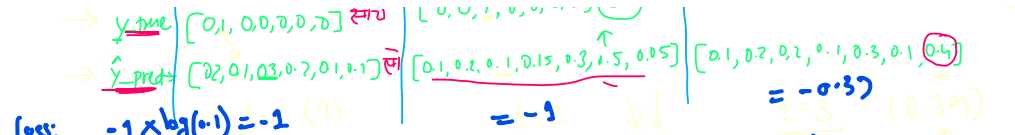


1st step থেকে output আসার কথা ছিল [0 1 0 0 0 0 0] যদি অন্য কোন output আসে তাহলে সেটা থেকে সঠিক value টাই আবার input দিতে হয়। এতে করে আমাদের convergence তাড়াতাড়ি হয়।

1st row:

	step 1	step 2	step 3
y_{true}	[0,1,0,0,0,0,0] সাথে	[0,0,1,0,0,0,0] এটা	[0,0,0,0,0,0,1] <end>
$\hat{y}_{pred}$	[0.2, 0.1, 0.3, 0.2, 0.1, 0.1]	[0.1, 0.2, 0.1, 0.15, 0.3, 0.5, 0.05]	[0.1, 0.2, 0.2, 0.1, 0.3, 0.1, 0.4]
	ভুল	ভুল	ঠিক

শুরুতে ভুল output এ আসবে কারণ শুরুতে random weight দিয়ে NN initialize করা থাকে।



* আমাদের forward propagation শেষ। এখন, আমরা loss calculation করবো। যেহেতু, আমাদের প্রবলেমটি হচ্ছে multi-class classification টাই, loss function হিসেবে categorical-cross-entrophy ব্যবহার করবো

$$\text{Categorical\_Cross\_entrophy} = -\sum_{1}^{7} y^{true} \log(y^{pred})$$

Total loss = -1 -1 -0.39 = -2.39 [লক্ষ্যনীয় যে, যার prediction সঠিক হয়েছে তার loss কম]

আমাদের, forward propagation শেষ, loss calculation শেষ। এখন, back propagation করতে হবে। আমরা আসি Back propagation দুই step এ হয়। ① gradient calculation ② update parameters.

1: Gradient calculation:

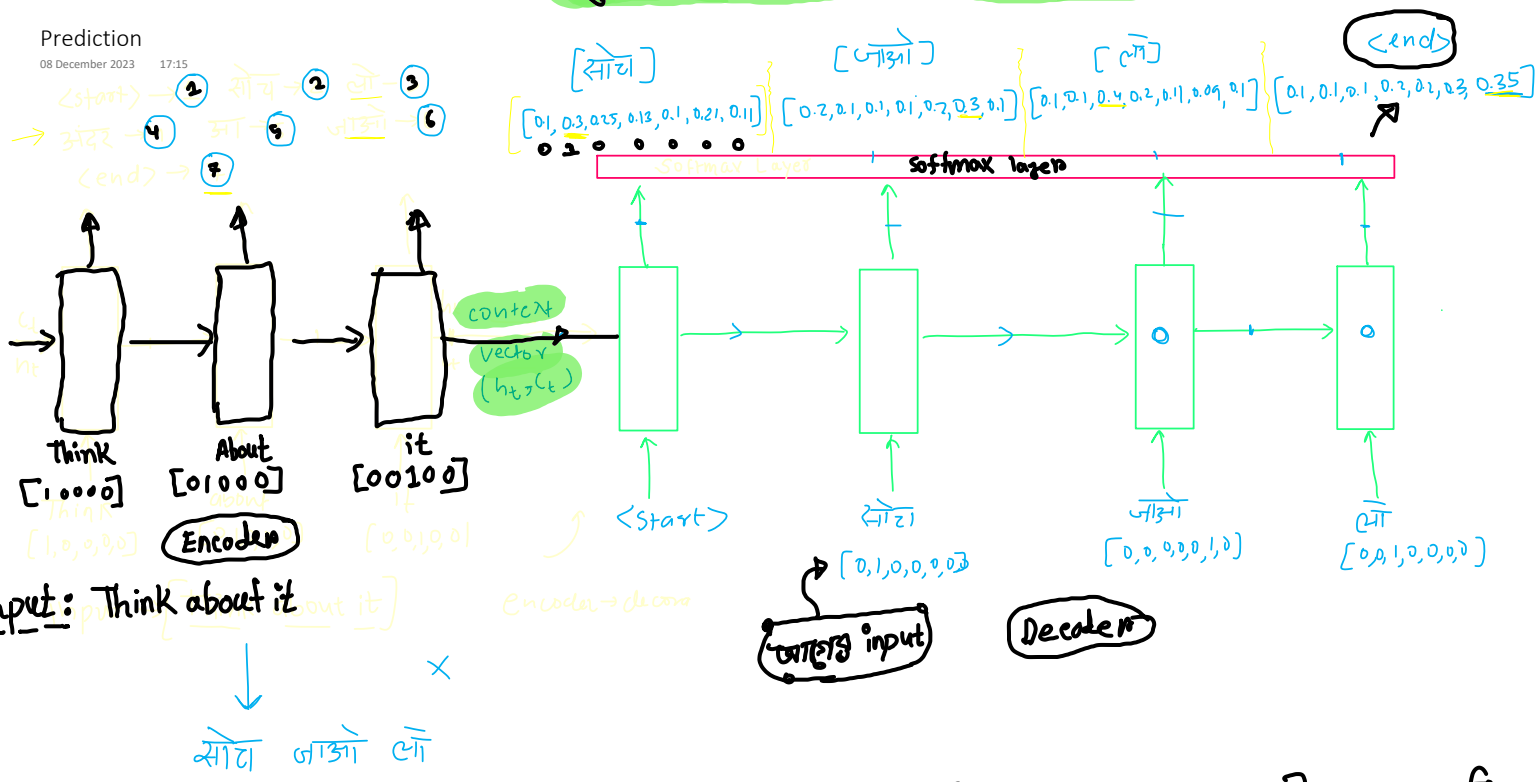
We calculate gradient of the loss with each of the trainable parameters.

এখানে, LSTM, Dense layer, Softmax এ যতগুলো parameter আছে সবগুলো দিয়ে gradient calculate করি। Gradient calculate করে loss function এ কোন parameter কতটুকু contribute করছে। আর সেই parameter কে কোন direction এ move করালে loss function এর loss কম হবে। এরপর সেই parameter update করি optimizer দিয়ে।

Training Complete হলে Prediction করবো

Prediction

08 December 2023 17:15



input: Think about it

↓

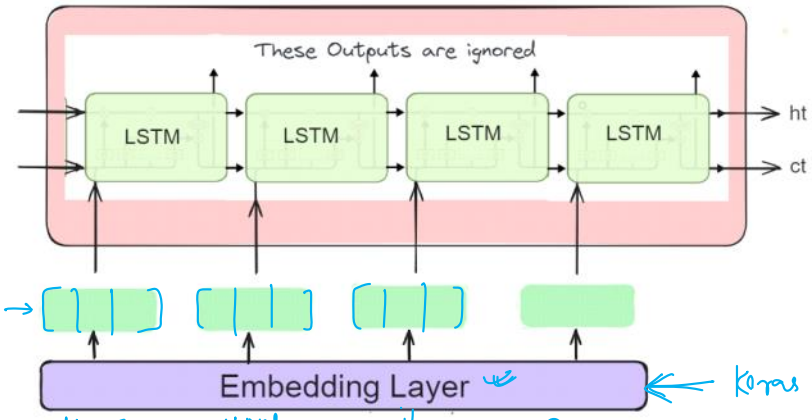
সোচা জাইও লো

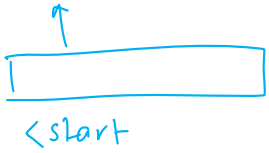
Prediction এর সময় Decoder এ আমাদের আগে কোন ডাটা নেই। এবার আমরা যেই আউটপুট পেয়েছি সেইটা next এর input হিসেবে দিয়ে দিচ্ছি। এখানে prediction টা ভুল। আমাদের training ভালো হয়নি। তবু এদের এমন হয়েছে।

Improvement of the encoder & decoder

Improvement 1 - Embeddings

08 December 2023 17:16





encoder
decoder

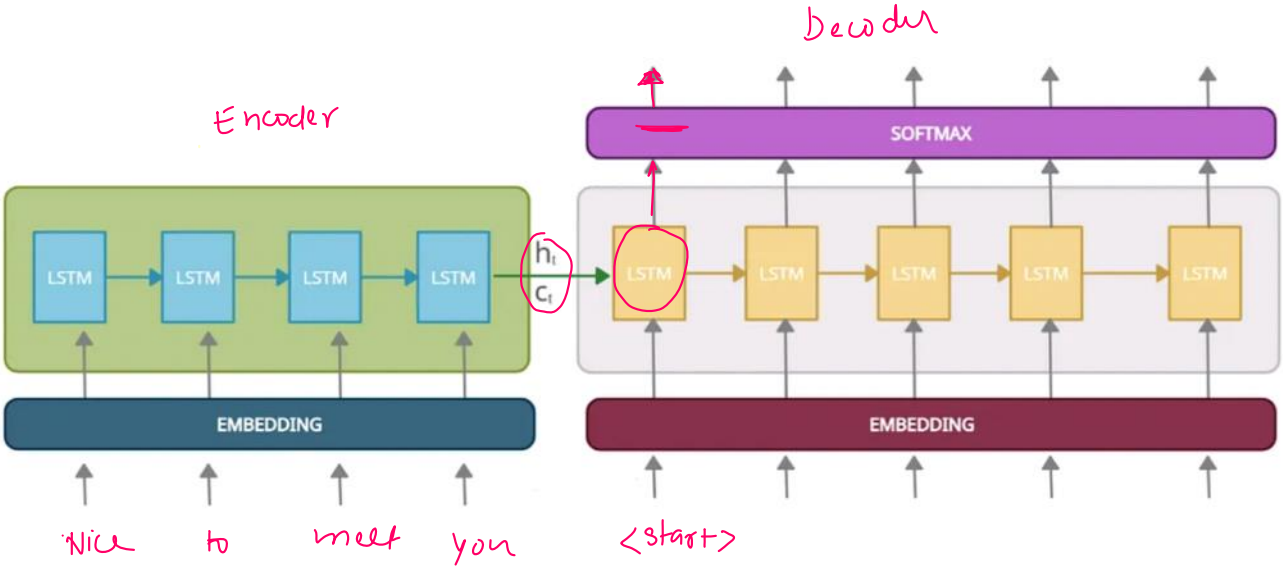
Keras

3 dim

low dimen / dense summary of word

pretrained → Word2vec / Glove

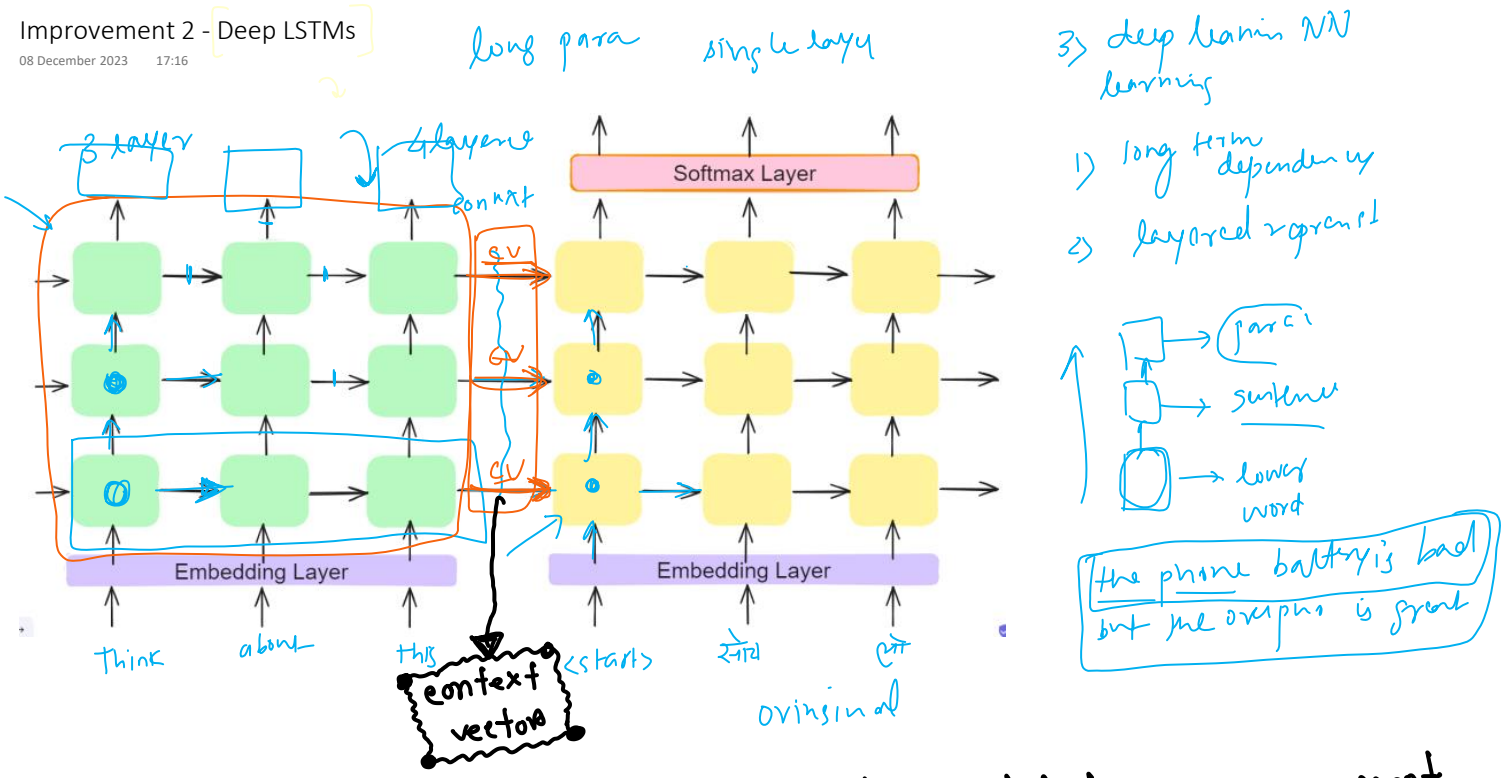
train



→ আমরা improvement হিসেবে embedding ব্যবহার করবো। Embedding technique আমরা আগে দেখেছি। এখানে, করতে আমরা embedding layer এ Integer encoded data দিয়ে একটা small dimention এর output পাই। যেমন আমরা 3 dimentional দিয়ে বুঝলাম। এখানে আমরা word এর summary পাচ্ছি। অর্থাৎ, embeding low dimention & Dense হয়। যেমন আমরা pre-trained embeding (Word2vec/Glove) ব্যবহার করতে পারি বা নিজের training dataset ব্যবহার করতে পারি। Now, see the final diagram of embeding both for in encoder and decoder।

Improvement 2 - Deep LSTMs

08 December 2023 17:16

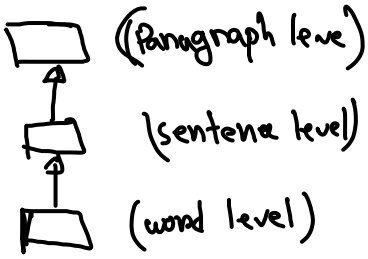


Multi-layer LSTM ব্যবহার করলে encoder-decoder Architecture-এ improvement আসে। Input দিলে embedding layer এর মধ্য দিয়ে LSTM-এ যায় তারপর, সেই LSTM তার পরের LSTM এর দিকে update করা ও উপরের Layer এর LSTM গুলোকে input provide করা।

Multi-layer LSTM ব্যবহার করার সুবিধা:

(i) Long term dependency গুলো handle করতে পারি। Long term dependency বলতে বোঝায় input যার word সংখ্যা অনেক বেশি বা বড় paragraph.

(ii) Layer representation বুঝতে পারে।

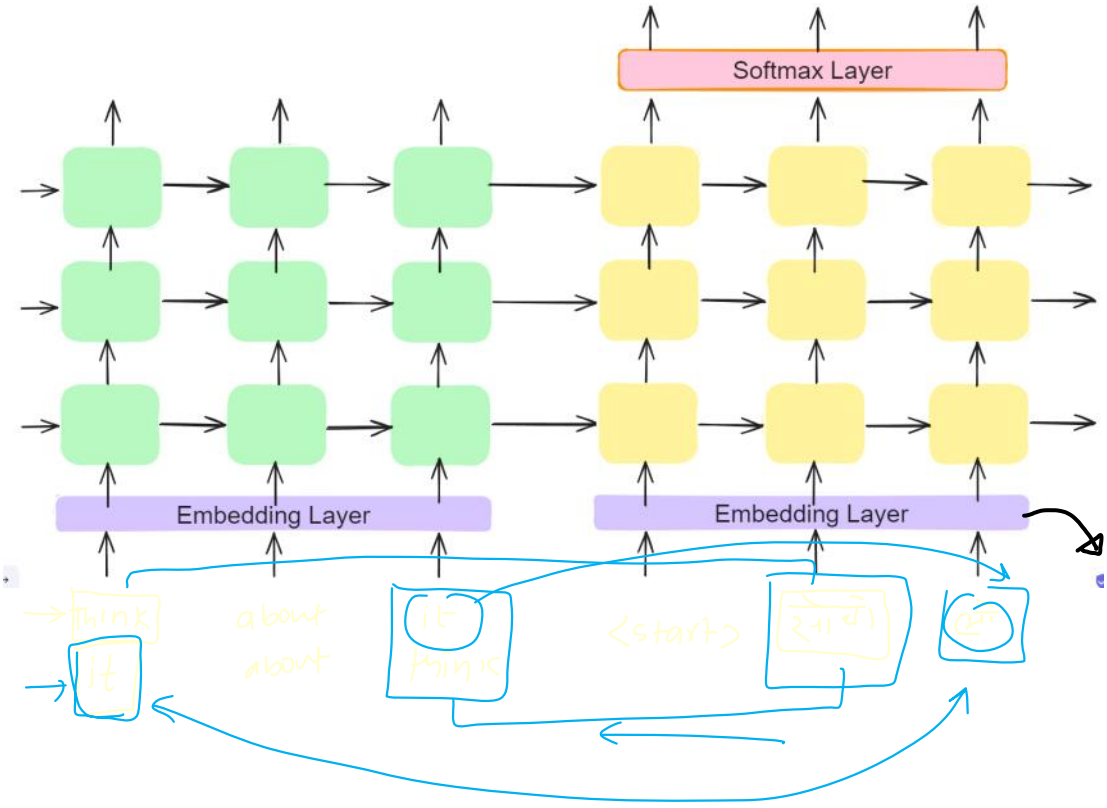


RNN এর মতো layer বেশি হলে semantic meaning ভালোভাবে capture করতে পারবে।

(iii) এখানে, আমরা Multi-layer LSTM ব্যবহার করছি এর কারণে আমাদের parameter সংখ্যা অনেক বৃদ্ধি পাচ্ছে। parameter এর সংখ্যা বৃদ্ধি পাওয়া মানে complex pattern কে capture করতে পারবে। আবার আবার overfitting করবিও না।

Improvement 3 - Reversing the Input

08 December 2023 17:16



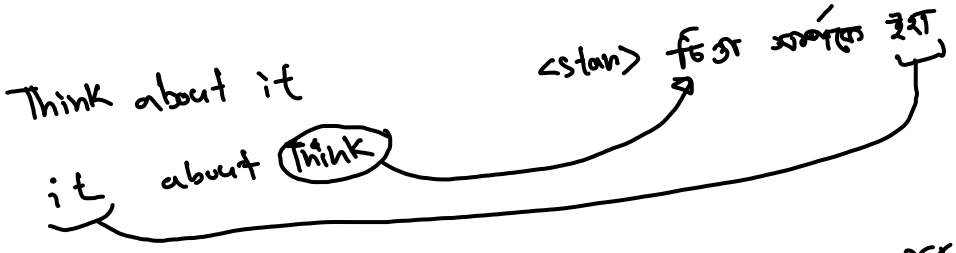
language
↓
initial context
eng - french

Input কে reverse করবো।

যদি input: Think about it.

reverse: It about think.

কিন্তু decoder এ কোন কিছু reverse হবে না



Think & first দুটি কাছাকাছি চলে গেছে reverse করার ফলে। As a result, gradient কে propagate করতে টাইম কম লাগছে। কিন্তু, it হয়ে গেছে দূরত্ব বেড়ে গেছে। মনে রাখতে হবে এটা অনেক language এর জন্য কাজ করে না। যদি এমন কোন language হতো Eg. sentence এর meaning অন্য language এ একটা word দিয়েই প্রকাশ করা যেতে সে ক্ষেত্রে এটা কাজে দিত।

The Sutskever Architecture (All these point will help us to understand the research paper)

08 December 2023 17:18

Application to Translation: The model focused on translating English to French, demonstrating the effectiveness of sequence-to-sequence learning in neural machine translation.

Special End-of-Sentence Symbol: Each sentence in the dataset was terminated with a unique end-of-sentence symbol ("<EOS>"), enabling the model to recognize the end of a sequence.

Dataset: The model was trained on a subset of 12 million sentences, comprising 348 million French words and 304 million English words, taken from a publicly available dataset.

Vocabulary Limitation: To manage computational complexity, fixed vocabularies for both languages were used, with 160,000 most frequent words for English and 80,000 for French. Words not in these vocabularies were replaced with a special "UNK" token.

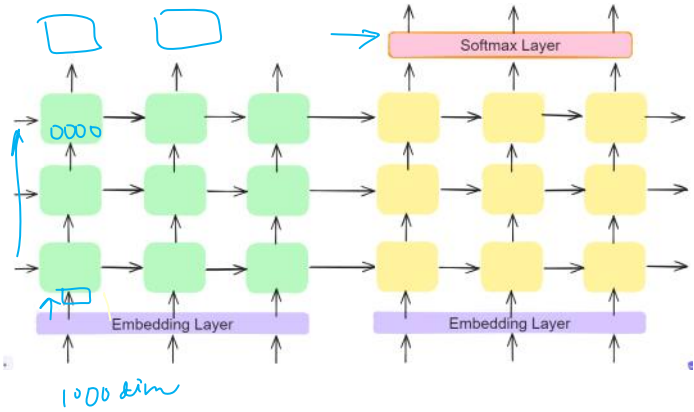
Reversing Input Sequences: The input sentences (English) were reversed before feeding them into the model, which was found to significantly improve the model's learning efficiency, especially for longer sentences.

Word Embeddings: The model used a 1000-dimensional word embedding layer to represent input words, providing dense, meaningful representations of each word.

Architecture Details: Both the input (encoder) and output (decoder) models had 4 layers, with each layer containing 1,000 units, showcasing a deep LSTM-based architecture.

Output Layer and Training: The output layer employed a Softmax function to generate the probability distribution over the target vocabulary. The model was trained end-to-end with these settings.

Performance - BLEU Score: The model achieved a BLEU score of 34.81, surpassing the baseline Statistical Machine Translation (SMT) system's score of 33.30 on the same dataset, marking a significant advancement in neural machine translation.



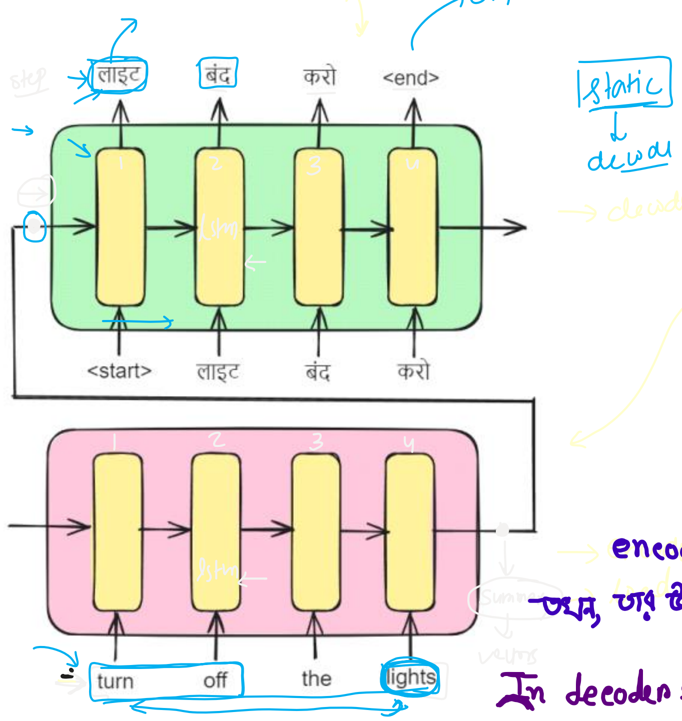
Attention Mechanism

Lecture : 69

improvement of encoder-decoder

The Why

20 December 2023 13:35



Encoder

Once upon a time in a small Indian village, a mischievous monkey stole a turban from a sleeping barber, wore it to a wedding, danced with the bewildered guests, accidentally got crowned the 'Banana King' by the local kids, and ended up leading a vibrant, impromptu parade of laughing villagers, cows, and street dogs, all while balancing a stack of mangoes on its head, creating a hilariously unforgettable spectacle and an amusing legend that the village still chuckles about every monsoon season.

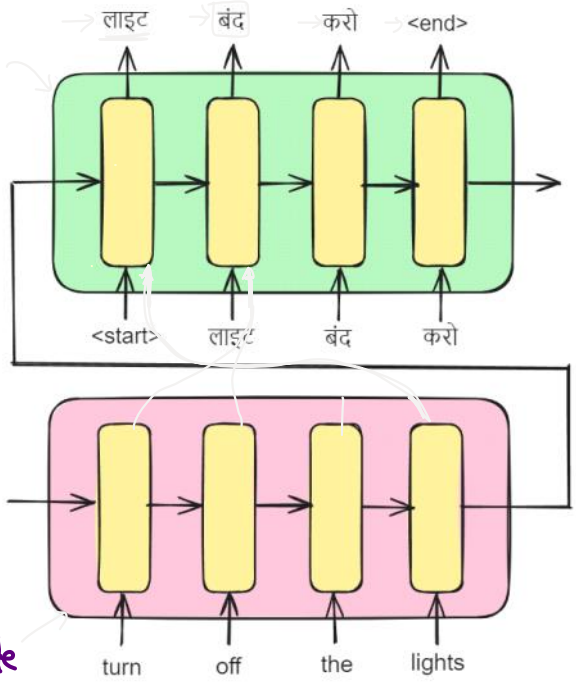
(এই গল্পটা ১ বার পড়ে চোখ বন্ধ করে অন্য language এ translation করতে পারবে? না কারণ এত বড় paragraph মনে রাখা সম্ভব না। আমাদের encoder-decoder সেম প্রবলেম ফেইস করে।)

encoder, এত বড় paragraph পড়ে যখন summary তৈরি করে তখন, তার উপর প্রেসার পড়ে as a human হিসেবে যেইটা আমাদের হয়েছিলো

In decoder side, প্রথম timestep এ লাইট এর translation এর জন্য light word টাই যথেষ্ট ছিল। পরের, word বন্ধ এর জন্য turn off এই দুইটায় যথেষ্ট ছিল। কিন্তু আমরা প্রত্যেক বার পুরো sentence টাই decoder এর কাছে পাঠাচ্ছি। এই ধরনের static representation এর জন্য decoder এর translation করতে প্রবলেম হয়। অর্থাৎ, যখন আমাদের লাইট এর English এর দরকার হচ্ছে তখন যদি পুরো sentence এর একটা পার্ট (lights) এ attention দিতে পারতাম তাহলে ভালো হতো। অর্থাৎ, static nature না হয়ে dynamic হলে ভালো হতো।

The Solution

20 December 2023 17:32



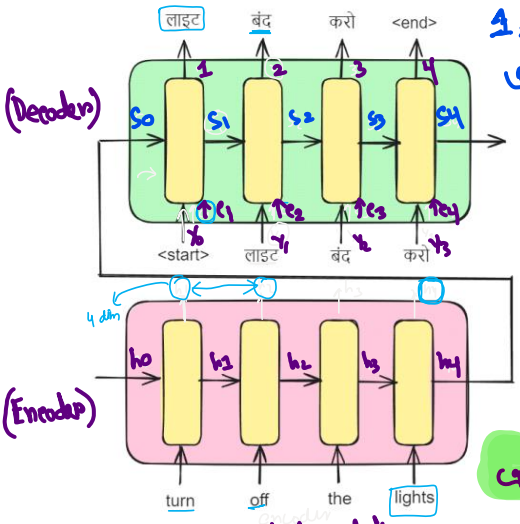
Once upon a time in a small Indian village, a mischievous monkey stole a turban from a sleeping barber, wore it to a wedding, danced with the bewildered guests, accidentally got crowned the 'Banana King' by the local kids, and ended up leading a vibrant, impromptu parade of laughing villagers, cows, and street dogs, all while balancing a stack of mangoes on its head, creating a hilariously unforgettable spectacle and an amusing legend that the village still chuckles about every monsoon season.

এর solution কি? আমরা human যখন কোন sentence translate করি তখন translation করে সামনের দিকে যাওয়ার সময় আমরা sentence এর একটা part কে focus করে translation করতে করতে যাই। ঠিক ছবির মতো অনেকটা। অর্থাৎ, light এর translation এর সময় decoder-কে বলতে হচ্ছে encoder এর 4 নং timestep টা গুরুত্বপূর্ণ। Decoder কে বুঝাতে হবে encoder এর কোনটা কোন set of timestep currently decoder এর জন্য গুরুত্বপূর্ণ। এই mechanism কে attention mechanism বলে।

What is attention Mechanism?

The What

21 December 2023 06:04



প্রথমে দুইটি দেখি, যেখানে, h হচ্ছে encoder এর hidden state. Y হচ্ছে decoder এ আগের LSTM এর output যেইটা current timestep এ input হিসেবে কাজ করবে। 1,2,3,4 হচ্ছে timestep. Normal, encoder-decoder Architecture এর decoder এ আমরা মনে করি timestep 2 এ, $(Y_1 \& S_1)$ LSTM input হিসেবে যেই।

Attention mechanism এ যে কোন time step এ আমাদের input হয় 3টা। (Y_2, S_1, C_2)

[C_2 হচ্ছে current timestep এ encoder এর কোন timestep টা important]

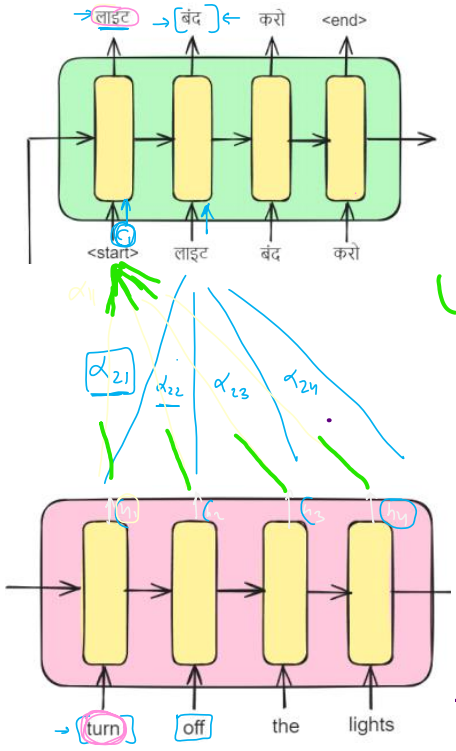
অর্থাৎ, যে কোন timestep এ আমরা C_i input দিচ্ছি যাকে attention input বলে। $[Y_{i-1}, S_{i-1}, C_i]$

এখন, e_{ij} কী যেখানে, → scalar, vector?

C_i → h_i

C_i হচ্ছে, current translation এর জন্য কোন encoder timestep important অর্থাৎ আমরা h_i কে pass করবো আরLSTM এ দেখেছি h_i হচ্ছে vector. C_i এর dimention h_i এর সমান হবে। আর 2টা h_i আটাচমেন্ট পেতে আমরা 2টার weighted sum করে পাঠাবো।

(যেখানে কিন্তু context vector নামে না)



decoder $i=1$

α_{11} α_{21}

udim

C_i → udim

scalar

$$C_1 = \alpha_{11} h_1 + \alpha_{12} h_2 + \alpha_{13} h_3 + \alpha_{14} h_4$$

↳ vectors

C_1 এ, $e_4 = 0.0$ অবশ্যে বেশি light কে গুরুত্ব দিবে।

C_4

$\alpha_{14} = 0.0$

$$C_2 = \alpha_{21} h_1 + \alpha_{22} h_2 + \alpha_{23} h_3 + \alpha_{24} h_4$$

encoder j

$$C_i = \sum_j \alpha_{ij} h_j$$

$i \times j = 16$ →

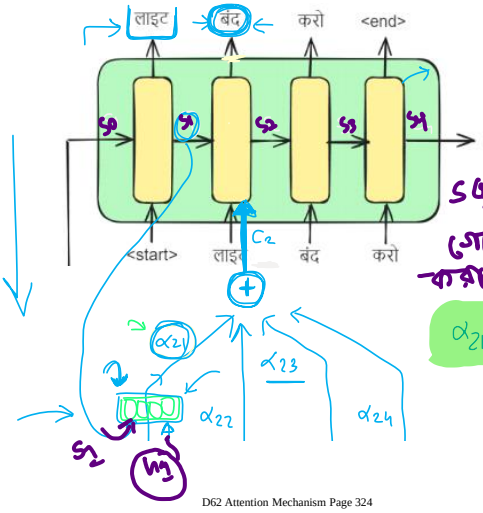
decoder এর timestep i

encoder " " j হয়,

$i=1$ এর জন্য j এর মান (1~4 হবে)।

total alphas = $i \times j = 4 \times 4 = 16$।

এখন, এই α কিভাবে calculation করা হয়?



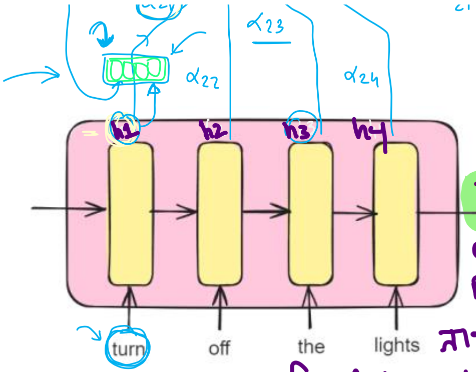
Let calculate, α_{21} এর আমরা alignment score বা similarity score বলি। α_{21} $(i=2, j=1)$ অর্থাৎ, α_{21} বুঝাচ্ছে, decoder এর timestep 2 এ, encoder এর timestep 1 এর ভূমিকা কতটুকু থাকবে? কিন্তু α_{21} এর মান S এর উপর dependent করে। আমরা জানতে পারি যে, যতটুকু translation হবে সেটার উপর ভিত্তি করে, h_1 এর কতটুকু ভূমিকা থাকবে timestep 2 print করাতে, অর্থাৎ,

$$\alpha_{21} = f(h_1, S_1)$$

ANN

$\alpha_{21} \rightarrow f(h_1, S_1)$

$\alpha_{23} \rightarrow f(h_3, S_1)$



$\alpha_{21} \to f(h_1, s_1)$

$[\alpha_{23} \to f(h_3, s_1)]$

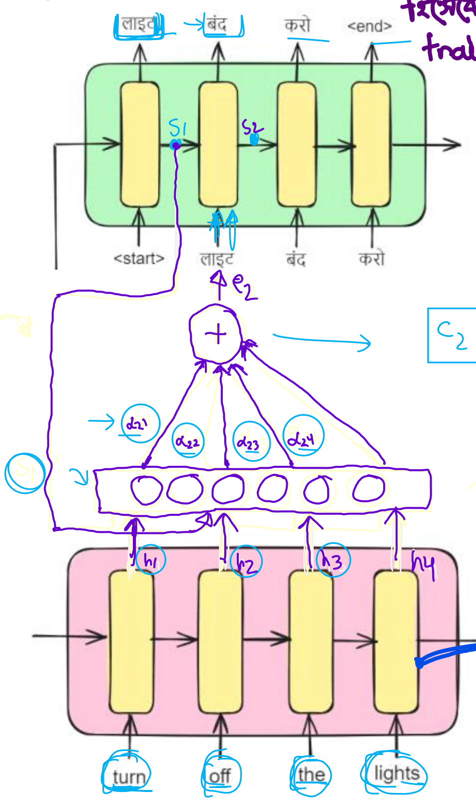
ANN

What function?

unique function approx.

In general, $\alpha_{ij} = f(h_j, s_{i-1})$

এখন, এই function টা কী হতে পারে। Researcher রা smart এই তারা function খুঁজতে ANN ব্যবহার করেছে। আমরা জানি, ANN, Universal function Appoximator। এর মানে হচ্ছে আমরা যদি enough data provide করি তাহলে, ANN যেকোন function এর appoximation হিসেবে কাজ করে। আমরা, ANN add করলে, input হিসেবে, α_{ij} এর ক্ষেত্রে h_2, s_1 যাবে আর যেহেতু ANN trainable তাই, encoder & decoder train হওয়ার সময় এইগুলোও হয়ে যাবে।



turn off the lights

লাইট α_{11} α_{13} α_{14}

বন্ধ α_{12} α_{21} α_{22}

করো

end

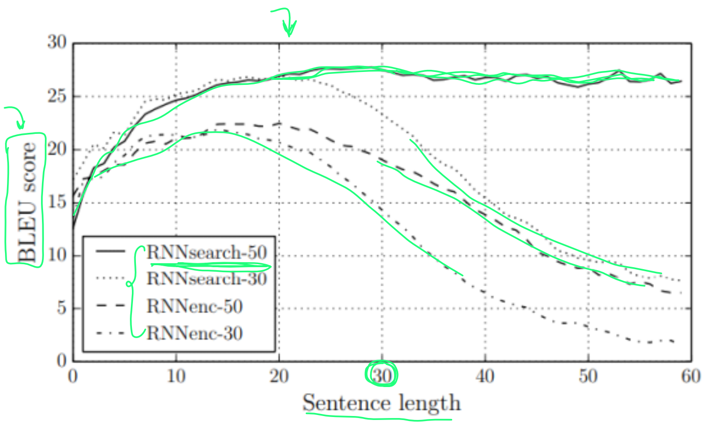
$c_2 = \alpha_{21}h_1 + \alpha_{22}h_2 + \alpha_{23}h_3 + \alpha_{24}h_4$

(e_2 calculation করার জন্য আমরা একবার s_1 ব্যবহার করেছি। আবার, s_2 calculation করার জন্যও আমাদের s_1 লাগবে।)

(Bidirection LSTM ব্যবহার করা হয়েছে।)

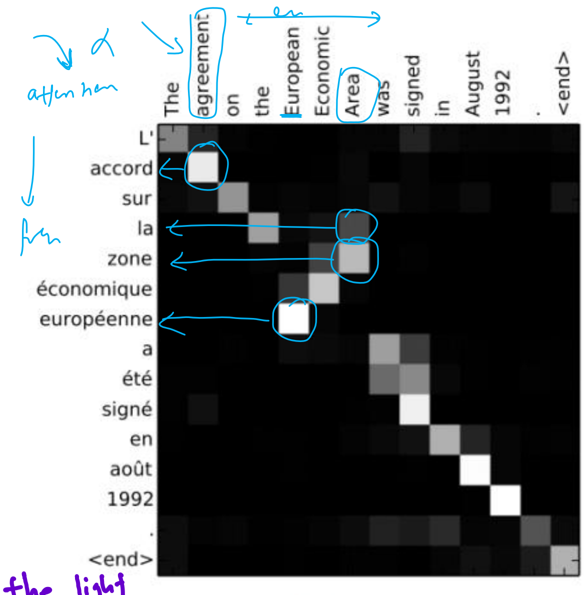
Bleu score 1 টা Matrics যার মাধ্যমে translation এর quality measure করা হয়।

eng-french



RNN-search-50 (Attention Mechanism)

This graph from official research paper.



(a)

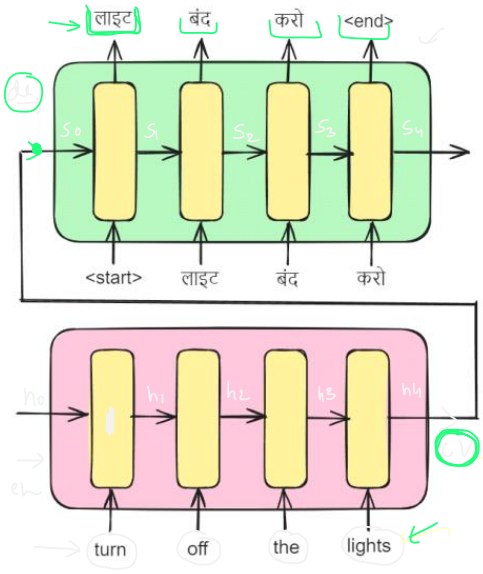
	turn	of	the	light
লাইট	α_{11}	α_{12}	α_{13}	α_{14}
বন্ধ	α_{21}	α_{22}	α_{23}	α_{24}
করো	α_{31}	α_{32}	α_{33}	α_{34}
end	α_{41}	α_{42}	α_{43}	α_{44}

→ এই value গুলো ব্যবহার করে heatmap graph বানাতে পারি।

Lecture -70

Recap

16 January 2024 16:10



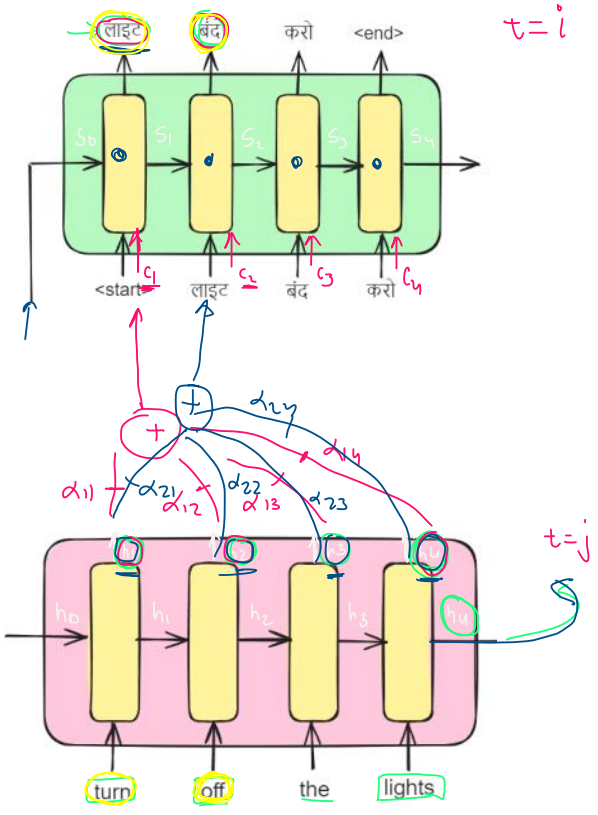
Encoder-Decoder

sentence > 30 words
paragraph
document

translation

bottleneck → Attention mechanism

Attention Mechanism



t=i

c_1 c_2 c_3 c_4

4 x 4 = 16

Weighted sum

$$c_i = \sum_{j=1}^{4} \alpha_{ij} h_j$$

α → alignment score

$c_1 = \alpha_{11} h_1 + \alpha_{12} h_2 + \alpha_{13} h_3 + \alpha_{14} h_4$

$c_2 = \alpha_{21} h_1 + \alpha_{22} h_2 + \alpha_{23} h_3 + \alpha_{24} h_4$

t=j

α → find out

- Bahdanau additive attention
- Luong attention

α find করার ২টা প্রসেস।

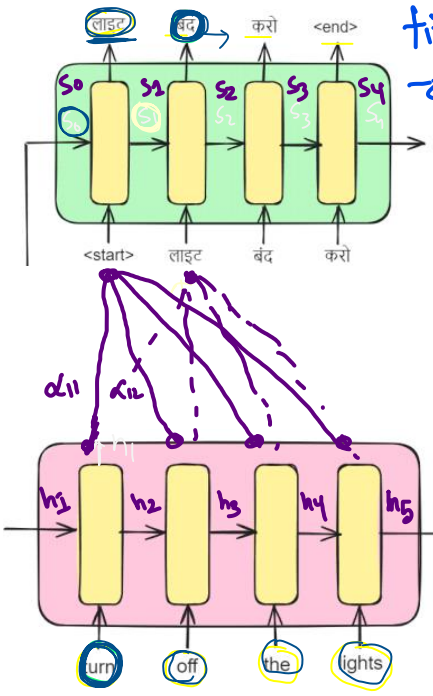
Bahdanau Attention

16 January 2024 16:11

α (attension score)

α (alignment score) = Decoder এর যেকোন particular timestep এর জন্য encoder এর hidden state গুলোর weighted কতটুকু হবে। (Based on previous hidden state + encoder timestep h.s.)

$c_i = \sum \alpha_{ij} h_j$ [i decoder এর timestep]

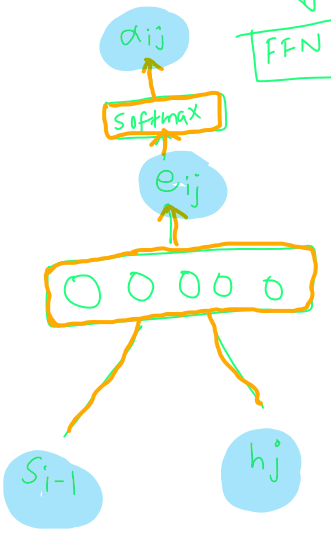


$\alpha_{11} = f(h_1 + s_0)$ $\alpha_{12} = f(h_2 + s_0)$

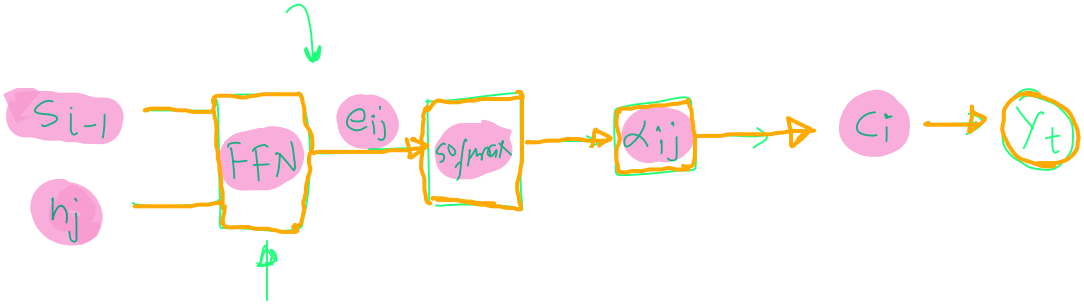
$\alpha_{21} = (h_1 + s_1)$ $\alpha_{22} = (h_2 + s_2)$

$\alpha_{ij} = f(h_j, s_{i-1})$

এখন, এই function কি হবে? Bahdanau এরা FFN (Feed Forward Network) ANN ব্যবহার করতে যেহেতু universal Function approximator হিসেবে ব্যবহৃত হয়।



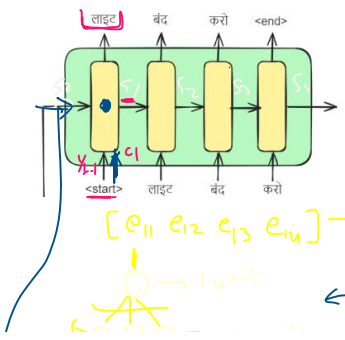
FFN → ANN → UFA



Flow diagram of Bahdanau

s_0 y_{t-1} c_1 → dshm → y_t (লাইট) $[s_1]$

$i=1$ $t=2$



$[e_{11}\ e_{12}\ e_{13}\ e_{14}]$ → softmax → $[\alpha_{11}\ \alpha_{12}\ \alpha_{13}\ \alpha_{14}]$

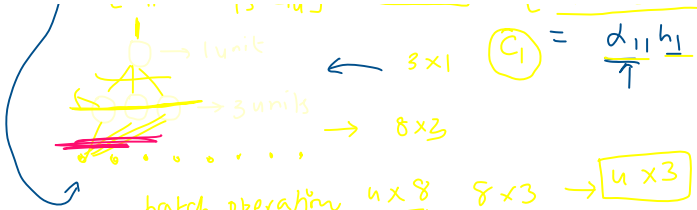
$c_1 = \sum \alpha_{1j} h_j$

$c_1 = \alpha_{11} h_1 + \alpha_{12} h_2 + \alpha_{13} h_3 + \alpha_{14} h_4$

3×1

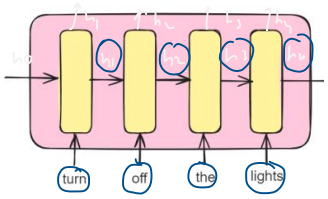
↳ $h_1\ h_2\ h_3\ h_4$

(s_{i-1}) h_j



$c_1 = \alpha_{11} h_1 + \alpha_{12} h_2 + \alpha_{13} h_3 + \alpha_{14} h_4$

α ($s_0 = [e\ f\ g\ h]$ $[s_{i-1} : h_j]$ $h_1\ h_2\ h_3\ h_4$



Vectors $h_0 = [a\ b\ c\ d]$ (4x3)(3x1)

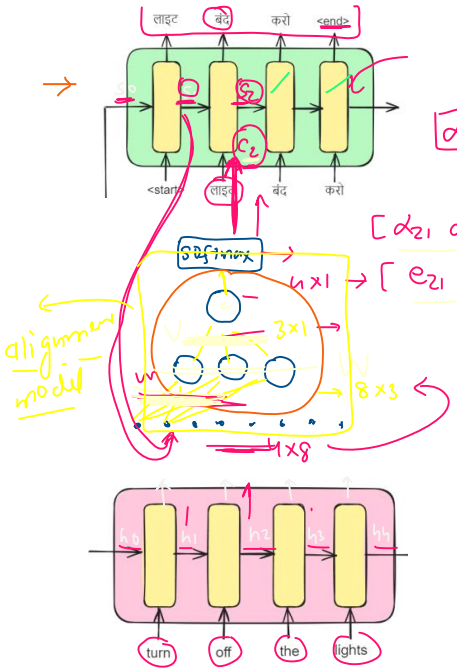
$\begin{bmatrix} e^{e_{11}} \\ \overline{e^{e_{11}} + e^{e_{12}} + e^{e_{13}} + e^{e_{14}}} \end{bmatrix}$ (4x1) → 4 numbers

4 rows / 8 cols → 8 cols

s_{01}	s_{02}	s_{03}	s_{04}	h_{11}	h_{12}	h_{13}	h_{14}
s_{01}	s_{02}	s_{03}	s_{04}	h_{21}	h_{22}	h_{23}	h_{24}
s_{01}	s_{02}	s_{03}	s_{04}	h_{31}	h_{32}	h_{33}	h_{34}
s_{01}	s_{02}	s_{03}	s_{04}	h_{41}	h_{42}	h_{43}	h_{44}

4 rows

time label



$c_2 = \alpha_{21} h_1 + \alpha_{22} h_2 + \alpha_{23} h_3 + \alpha_{24} h_4$

α

$[\alpha_{21}\ \alpha_{22}\ \alpha_{23}\ \alpha_{24}]$ — $[e_{21}\ e_{22}\ e_{23}\ e_{24}]$ 4x3

s_{11}	s_{12}	s_{13}	s_{14}	h_{11}	h_{12}	h_{13}	h_{14}
s_{11}	s_{12}	s_{13}	s_{14}	h_{21}	h_{22}	h_{23}	h_{24}
s_{11}	s_{12}	s_{13}	s_{14}	h_{31}	h_{32}	h_{33}	h_{34}
s_{11}	s_{12}	s_{13}	s_{14}	h_{41}	h_{42}	h_{43}	h_{44}

4x8

time distributed FNN

$c_{ij} = \sum \alpha_{ij} h_j$

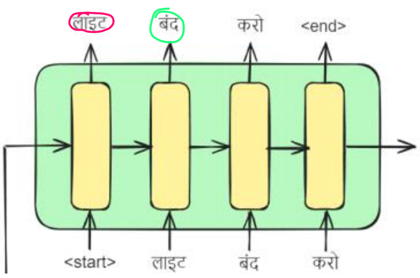
$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T} \exp(e_{ik})}$

$e_{ij} = v_a^T \tanh(W[s_{i-1} : h_j] + b)$

additive attention

Luong Attention, Bahdanau এর থেকে এসেছে অনেক বেশি improve আছে।

Luong Attention
17 January 2024 00:09



parameters → slow

$c_i = \sum \alpha_{ij} h_j$ → FFN

$\alpha [V \tan(W[s_{i-1}; h_j]+b)]$ (T; transpose)

Bahdanau ~ $\alpha_{ij} = f(s_{i-1}, h_j)$ ×

Luong ~ $\alpha_{ij} = f(s_i, h_j)$ → $[s_i^T . h_j]$ → dot product

current — ① diff — updated info — dynamic — adjust

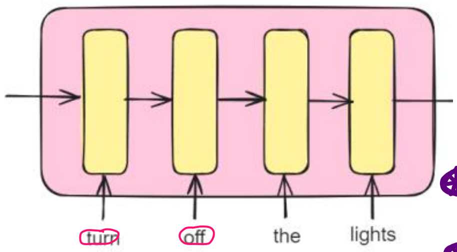
$s_i = [a\ b\ c\ d]$

$h_j = [e\ f\ g\ h]$

s_i^T $\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$ h_j $[e\ f\ g\ h]$

$[ae + bf + cg + dh]$ → scalar → attention

e_{ij} → softmax → α_{ij}

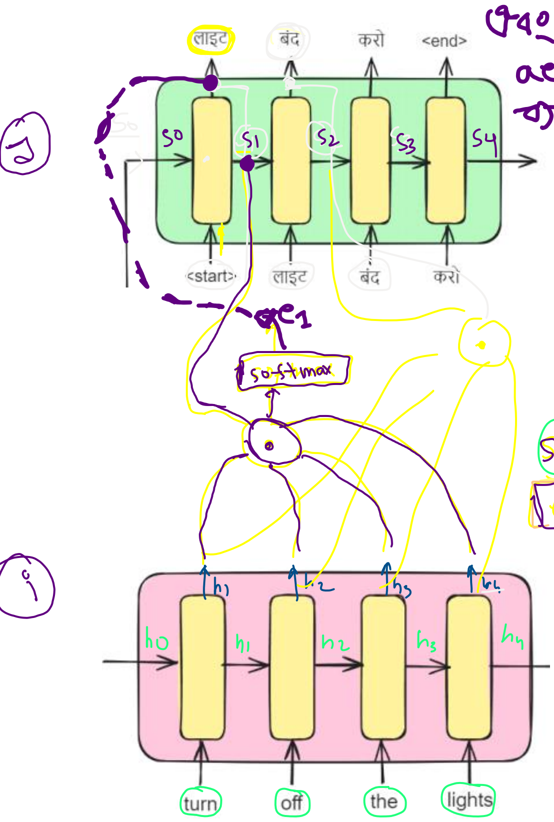


⊛ Bahdanau তে decoder এর previous hidden state (s_{i-1}) ব্যবহৃত হয়

⊛ Luong " " " current " " (s_i) " "

⊛ " " FFN এর পরিবর্তে dot product ব্যবহৃত হয়।

আমরা current hidden state ব্যবহার করার ফলে up to date info পাচ্ছি এবং এটা আরো dynamic হচ্ছে nature এ। আর এর ফলে Luong এর accuracy ও বেড়েছে bahdanau এর তুলনায়। Dot product ব্যবহার করার ফলে এটা fast ও হয়ে যাচ্ছে।



$[e_{21}\ e_{22}\ e_{23}\ e_{24}]$ softmax → $\alpha_{21}\ \alpha_{22}\ \alpha_{23}\ \alpha_{24}$ → c_2

s_1h_1 s_1h_2 s_1h_3 s_1h_4

$[e_{11}\ e_{12}\ e_{13}\ e_{14}]$

$\alpha_{11}\ \alpha_{12}\ \alpha_{13}\ \alpha_{14}$ → c_1

$c_i = \sum \alpha_{ij} h_j$

এখানে, s_0 এর পরিবর্তে ($s_i h_i$) dot product করবো। এর থেকে e_{ij} পাবো, এরপর softmax layer এ দিয়ে c_i calculate করবো।

Bahdanau তে s ছিল previous hidden state কিন্তু এইক্ষেত্রে current hidden state তাই, আমরা c_1 কে s_0 এর output এর সাথে add করবো। এখানে, c_1 add করবো s_0 এর সাথে তার পর আবার, softmax লাগাবো then 1st word টা print করবো।