Title: Seq2seq

URL: https://en.wikipedia.org/wiki/Seq2seq

PageID: 62607005

Categories: Category:2014 in artificial intelligence, Category:Artificial neural networks, Category:Natural language processing

-----

Seq2seq is a family of machine learning approaches used for natural language processing . [1] Applications include language translation , [2] image captioning , [3] conversational models, [4] speech recognition , [5] and text summarization . [6] Seq2seq uses sequence transformation : it turns one sequence into another sequence.

History

One naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say: 'This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.

— Warren Weaver , Letter to Norbert Wiener , March 4, 1947

seq2seq is an approach to machine translation (or more generally, sequence transduction) with roots in information theory, where communication is understood as an encode-transmit-decode process, and machine translation can be studied as a special case of communication. This viewpoint was elaborated, for example, in the noisy channel model of machine translation.

In practice, seq2seq maps an input sequence into a real-numerical vector by using a neural network (the encoder), and then maps it back to an output sequence using another neural network (the decoder).

The idea of encoder-decoder sequence transduction had been developed in the early 2010s (see [2] [1] for previous papers). The papers most commonly cited as the originators that produced seq2seq are two papers from 2014. [2] [1]

In the seq2seq as proposed by them, both the encoder and the decoder were LSTMs . This had the "bottleneck" problem, since the encoding vector has a fixed size, so for long input sequences, information would tend to be lost, as they are difficult to fit into the fixed-length encoding vector. The attention mechanism , proposed in 2014, [7] resolved the bottleneck problem. They called their model RNNsearch , as it "emulates searching through a source sentence during decoding a translation".

A problem with seq2seq models at this point was that recurrent neural networks are difficult to parallelize. The 2017 publication of Transformers [8] resolved the problem by replacing the encoding RNN with self-attention Transformer blocks ("encoder blocks"), and the decoding RNN with cross-attention causally-masked Transformer blocks ("decoder blocks").

Priority dispute

One of the papers cited as the originator for seq2seq is (Sutskever et al 2014), [1] published at Google Brain while they were on Google's machine translation project. The research allowed Google to overhaul Google Translate into Google Neural Machine Translation in 2016. [1] [9] Tomáš Mikolov claims to have developed the idea (before joining Google Brain) of using a "neural language model on pairs of sentences... and then [generating] translation after seeing the first sentence"—which he equates with seq2seq machine translation, and to have mentioned the idea to Ilya Sutskever and Quoc Le (while at Google Brain), who failed to acknowledge him in their paper. [10] Mikolov had worked on RNNLM (using RNN for language modelling) for his PhD thesis, [11] and is more notable for developing word2vec .

Architecture

The main reference for this section is. [12]

Encoder

The encoder is responsible for processing the input sequence and capturing its essential information, which is stored as the hidden state of the network and, in a model with attention mechanism, a context vector. The context vector is the weighted sum of the input hidden states and is generated for every time instance in the output sequences.

Decoder

The decoder takes the context vector and hidden states from the encoder and generates the final output sequence. The decoder operates in an autoregressive manner, producing one element of the output sequence at a time. At each step, it considers the previously generated elements, the context vector, and the input sequence information to make predictions for the next element in the output sequence. Specifically, in a model with attention mechanism, the context vector and the hidden state are concatenated together to form an attention hidden vector, which is used as an input for the decoder.

The seq2seq method developed in the early 2010s uses two neural networks: an encoder network converts an input sentence into numerical vectors, and a decoder network converts those vectors to sentences in the target language. The Attention mechanism was grafted onto this structure in 2014 and shown below. Later it was refined into the encoder-decoder Transformer architecture of 2017.

Training vs prediction

There is a subtle difference between training and prediction. During training time, both the input and the output sequences are known. During prediction time, only the input sequence is known, and the output sequence must be decoded by the network itself.

Specifically, consider an input sequence x 1 : n {\displaystyle x_{1:n}} and output sequence y 1 : m {\displaystyle y_{1:m}} . The encoder would process the input x 1 : n {\displaystyle x_{1:n}} step by step. After that, the decoder would take the output from the encoder, as well as the as input, and produce a prediction y ^ 1 {\displaystyle {\hat {y}}_{1}} . Now, the question is: what should be input to the decoder in the next step?

A standard method for training is "teacher forcing". In teacher forcing, no matter what is output by the decoder, the next input to the decoder is always the reference. That is, even if y ^ 1 ≠ y 1 {\displaystyle {\hat {y}}_{1}\neq y_{1}} , the next input to the decoder is still y 1 {\displaystyle y_{1}} , and so on.

During prediction time, the "teacher" y 1 : m {\displaystyle y_{1:m}} would be unavailable. Therefore, the input to the decoder must be y ^ 1 {\displaystyle {\hat {y}}_{1}} , then y ^ 2 {\displaystyle {\hat {y}}_{2}} , and so on.

It is found that if a model is trained purely by teacher forcing, its performance would degrade during prediction time, since generation based on the model's own output is different from generation based on the teacher's output. This is called exposure bias or a train/test distribution shift . A 2015 paper recommends that, during training, randomly switch between teacher forcing and no teacher forcing. [13]

Attention for seq2seq

The attention mechanism is an enhancement introduced by Bahdanau et al. in 2014 to address limitations in the basic Seq2Seq architecture where a longer input sequence results in the hidden state output of the encoder becoming irrelevant for the decoder. It enables the model to selectively focus on different parts of the input sequence during the decoding process. At each decoder step, an alignment model calculates the attention score using the current decoder state and all of the attention hidden vectors as input. An alignment model is another neural network model that is trained jointly with the seq2seq model used to calculate how well an input, represented by the hidden state, matches with the previous output, represented by attention hidden state. A softmax function is then applied to the attention score to get the attention weight.

In some models, the encoder states are directly fed into an activation function, removing the need for alignment model. An activation function receives one decoder state and one encoder state and returns a scalar value of their relevance.

Consider the seq2seq language English-to-French translation task. To be concrete, let us consider the translation of "the zone of international control ", which should translate to "la zone de contrôle international ". Here, we use the special token as a control character to delimit the end of input for both the encoder and the decoder.

An input sequence of text x 0 , x 1 , … {\displaystyle x_{0},x_{1},\dots } is processed by a neural network (which can be an LSTM, a Transformer encoder, or some other network) into a sequence of real-valued vectors h 0 , h 1 , … {\displaystyle h_{0},h_{1},\dots } , where h {\displaystyle h} stands for "hidden vector".

After the encoder has finished processing, the decoder starts operating over the hidden vectors, to produce an output sequence y 0 , y 1 , … {\displaystyle y_{0},y_{1},\dots } , autoregressively. That is, it always takes as input both the hidden vectors produced by the encoder, and what the decoder itself has produced before, to produce the next output word:

(h 0 , h 1 , … {\displaystyle h_{0},h_{1},\dots } , "") → "la"

(h 0 , h 1 , … {\displaystyle h_{0},h_{1},\dots } , " la") → "la zone"

(h 0 , h 1 , … {\displaystyle h_{0},h_{1},\dots } , " la zone") → "la zone de"

...

(h 0 , h 1 , … {\displaystyle h_{0},h_{1},\dots } , " la zone de contrôle international") → "la zone de contrôle international "

Here, we use the special token as a control character to delimit the start of input for the decoder. The decoding terminates as soon as "" appears in the decoder output.

Attention weights

As hand-crafting weights defeats the purpose of machine learning, the model must compute the attention weights on its own. Taking analogy from the language of database queries , we make the model construct a triple of vectors: key, query, and value. The rough idea is that we have a "database" in the form of a list of key-value pairs. The decoder sends in a query , and obtains a reply in the form of a weighted sum of the values , where the weight is proportional to how closely the query resembles each key .

The decoder first processes the "" input partially, to obtain an intermediate vector h 0 d {\displaystyle h_{0}^{d}} , the 0th hidden vector of decoder. Then, the intermediate vector is transformed by a linear map W Q {\displaystyle W^{Q}} into a query vector q 0 = h 0 d W Q {\displaystyle q_{0}=h_{0}^{d}W^{Q}} . Meanwhile, the hidden vectors outputted by the encoder are transformed by another linear map W K {\displaystyle W^{K}} into key vectors k 0 = h 0 W K , k 1 = h 1 W K , … {\displaystyle k_{0}=h_{0}W^{K},k_{1}=h_{1}W^{K},\dots } . The linear maps are useful for providing the model with enough freedom to find the best way to represent the data.

Now, the query and keys are compared by taking dot products: q 0 k 0 T , q 0 k 1 T , … {\displaystyle q_{0}k_{0}^{T},q_{0}k_{1}^{T},\dots } . Ideally, the model should have learned to compute the keys and values, such that q 0 k 0 T {\displaystyle q_{0}k_{0}^{T}} is large, q 0 k 1 T {\displaystyle q_{0}k_{1}^{T}} is small, and the rest are very small. This can be interpreted as saying that the attention weight should be mostly applied to the 0th hidden vector of the encoder, a little to the 1st, and essentially none to the rest.

In order to make a properly weighted sum, we need to transform this list of dot products into a probability distribution over 0 , 1 , … {\displaystyle 0,1,\dots } . This can be accomplished by the softmax function , thus giving us the attention weights: (w 00 , w 01 , …) = s o f t m a x (q 0 k 0 T , q 0 k 1 T , …) {\displaystyle (w_{00},w_{01},\dots)=\mathrm {softmax} (q_{0}k_{0}^{T},q_{0}k_{1}^{T},\dots)} This is then used to compute the context vector : c 0 = w 00 v 0 + w 01 v 1 + ■ {\displaystyle c_{0}=w_{00}v_{0}+w_{01}v_{1}+\cdots } where v 0 = h 0 W V , v 1 = h 1 W V , … {\displaystyle v_{0}=h_{0}W^{V},v_{1}=h_{1}W^{V},\dots } are the value vectors, linearly transformed by another matrix to provide the model with freedom to find the best way to represent values. Without the matrices W Q , W K , W V {\displaystyle W^{Q},W^{K},W^{V}} , the model would be forced to use the same hidden vector for both key and value, which might not be appropriate, as these two tasks are not the same.

This is the dot-attention mechanism. The particular version described in this section is "decoder cross-attention", as the output context vector is used by the decoder, and the input keys and values come from the encoder, but the query comes from the decoder, thus "cross-attention".

More succinctly, we can write it as c 0 = A t t e n t i o n (h 0 d W Q , H W K , H W V) = s o f t m a x ((h 0 d W Q) (H W K) T) (H W V) {\displaystyle c_{0}=\mathrm {Attention} (h_{0}^{d}W^{Q},HW^{K},HW^{V})=\mathrm {softmax} ((h_{0}^{d}W^{Q})\;(HW^{K})^{T})(HW^{V})} where the matrix H {\displaystyle H} is the matrix whose rows are h 0 , h 1 , … {\displaystyle h_{0},h_{1},\dots } . Note that the querying vector, h 0 d {\displaystyle h_{0}^{d}} , is not necessarily the same as the key-value vector h 0 {\displaystyle h_{0}} . In fact, it is theoretically possible for query, key, and value vectors to all be different, though that is rarely done in practice.

Other applications

In 2019, Facebook announced its use in symbolic integration and resolution of differential equations . The company claimed that it could solve complex equations more rapidly and with greater accuracy than commercial solutions such as Mathematica , MATLAB and Maple . First, the equation is parsed into a tree structure to avoid notational idiosyncrasies. An LSTM neural network then applies its standard pattern recognition facilities to process the tree. [14] [15]

In 2020, Google released Meena, a 2.6 billion parameter seq2seq-based chatbot trained on a 341 GB data set. Google claimed that the chatbot has 1.7 times greater model capacity than OpenAI 's GPT-2 . [4]

In 2022, Amazon introduced AlexaTM 20B, a moderate-sized (20 billion parameter) seq2seq language model . It uses an encoder-decoder to accomplish few-shot learning. The encoder outputs a representation of the input that the decoder uses as input to perform a specific task, such as translating the input into another language. The model outperforms the much larger GPT-3 in language translation and summarization. Training mixes denoising (appropriately inserting missing text in strings) and causal-language-modeling (meaningfully extending an input text). It allows adding features across different languages without massive training workflows. AlexaTM 20B achieved state-of-the-art performance in few-shot-learning tasks across all Flores-101 language pairs, outperforming GPT-3 on several tasks. [16]

See also

Artificial neural network

References

External links

Voita, Lena. "Sequence to Sequence (seq2seq) and Attention" . Retrieved 2023-12-20 .

"A ten-minute introduction to sequence-to-sequence learning in Keras" . blog.keras.io . Retrieved 2019-12-19 .

v

t

e

AI-complete

Bag-of-words

n -gram Bigram Trigram

Bigram

Trigram

Computational linguistics

Natural language understanding

Stop words

Text processing

Argument mining

Collocation extraction

Concept mining

Coreference resolution

Deep linguistic processing

Distant reading

Information extraction

Named-entity recognition

Ontology learning

Parsing Semantic parsing Syntactic parsing

Semantic parsing

Syntactic parsing

Part-of-speech tagging

Semantic analysis

Semantic role labeling

Semantic decomposition

Semantic similarity

Sentiment analysis

Terminology extraction

Text mining

Textual entailment

Truecasing

Word-sense disambiguation

Word-sense induction

Compound-term processing

Lemmatisation

Lexical analysis

Text chunking

Stemming

Sentence segmentation

Word segmentation

Multi-document summarization

Sentence extraction

Text simplification
Computer-assisted
Example-based
Rule-based
Statistical
Transfer-based
Neural
BERT
Document-term matrix
Explicit semantic analysis
fastText
GloVe
Language model (large)
Latent semantic analysis
Seq2seq
Word embedding
Word2vec
Corpus linguistics
Lexical resource
Linguistic Linked Open Data
Machine-readable dictionary
Parallel text
PropBank
Semantic network
Simple Knowledge Organization System
Speech corpus
Text corpus
Thesaurus (information retrieval)
Treebank
Universal Dependencies
BabelNet
Bank of English
DBpedia
FrameNet
Google Ngram Viewer
UBY
WordNet
Wikidata

Speech recognition

Speech segmentation

Speech synthesis

Natural language generation

Optical character recognition

Document classification

Latent Dirichlet allocation

Pachinko allocation

Automated essay scoring

Concordancer

Grammar checker

Predictive text

Pronunciation assessment

Spell checker

Chatbot

Interactive fiction

Question answering

Virtual assistant

Voice user interface

Formal semantics

Hallucination

Natural Language Toolkit

spaCy

v

t

e

History timeline

timeline

Companies

Projects

Parameter Hyperparameter

Hyperparameter

Loss functions

Regression Bias–variance tradeoff Double descent Overfitting

Bias–variance tradeoff

Double descent

Overfitting

Clustering

Gradient descent SGD Quasi-Newton method Conjugate gradient method

SGD

Quasi-Newton method

Conjugate gradient method

Backpropagation

Attention

Convolution

Normalization Batchnorm

Batchnorm

Activation Softmax Sigmoid Rectifier

Softmax

Sigmoid

Rectifier

Gating

Weight initialization

Regularization

Datasets Augmentation

Augmentation

Prompt engineering

Reinforcement learning Q-learning SARSA Imitation Policy gradient

Q-learning

SARSA

Imitation

Policy gradient

Diffusion

Latent diffusion model

Autoregression

Adversary

RAG

Uncanny valley

RLHF

Self-supervised learning

Reflection

Recursive self-improvement

Hallucination

Word embedding

Vibe coding

Machine learning In-context learning

In-context learning

Artificial neural network Deep learning

Deep learning

Language model Large language model NMT

Large language model

NMT

Reasoning language model

Model Context Protocol

Intelligent agent

Artificial human companion

Humanity's Last Exam

Artificial general intelligence (AGI)

AlexNet

WaveNet

Human image synthesis

HWR

OCR

Computer vision

Speech synthesis 15.ai ElevenLabs

15.ai

ElevenLabs

Speech recognition Whisper

Whisper

Facial recognition

AlphaFold

Text-to-image models Aurora DALL-E Firefly Flux Ideogram Imagen Midjourney Recraft Stable Diffusion

Aurora

DALL-E

Firefly

Flux

Ideogram

Imagen

Midjourney

Recraft

Stable Diffusion

Text-to-video models Dream Machine Runway Gen Hailuo AI Kling Sora Veo

Dream Machine

Runway Gen

Hailuo AI

Kling

Sora

Veo

Music generation Riffusion Suno AI Udio

Riffusion

Suno AI

Udio

Word2vec

Seq2seq

GloVe

BERT

T5

Llama

Chinchilla AI

PaLM

GPT 1 2 3 J ChatGPT 4 4o o1 o3 4.5 4.1 o4-mini 5

1

2

3

J

ChatGPT

4

4o

o1

o3

4.5

4.1

o4-mini

5

Claude

Gemini Gemini (language model) Gemma

Gemini (language model)

Gemma

Grok

LaMDA

BLOOM

DBRX

Project Debater

IBM Watson

IBM Watsonx

Granite

PanGu-Σ

DeepSeek

Qwen

AlphaGo

AlphaZero

OpenAI Five

Self-driving car

MuZero

Action selection AutoGPT

AutoGPT

Robot control

Alan Turing

Warren Sturgis McCulloch

Walter Pitts

John von Neumann

Claude Shannon

Shun'ichi Amari

Kunihiko Fukushima

Takeo Kanade

Marvin Minsky

John McCarthy

Nathaniel Rochester

Allen Newell

Cliff Shaw

Herbert A. Simon

Oliver Selfridge

Frank Rosenblatt

Bernard Widrow

Joseph Weizenbaum

Seymour Papert

Seppo Linnainmaa

Paul Werbos

Geoffrey Hinton

John Hopfield

Jürgen Schmidhuber

Yann LeCun

Yoshua Bengio

Lotfi A. Zadeh

Stephen Grossberg

Alex Graves

James Goodnight

Andrew Ng

Fei-Fei Li

Alex Krizhevsky

Ilya Sutskever

Oriol Vinyals

Quoc V. Le

Ian Goodfellow

Demis Hassabis

David Silver

Andrej Karpathy

Ashish Vaswani

Noam Shazeer

Aidan Gomez

John Schulman

Mustafa Suleyman

Jan Leike

Daniel Kokotajlo

François Chollet

Neural Turing machine

Differentiable neural computer

Transformer Vision transformer (ViT)

Vision transformer (ViT)

Recurrent neural network (RNN)

Long short-term memory (LSTM)

Gated recurrent unit (GRU)

Echo state network

Multilayer perceptron (MLP)

Convolutional neural network (CNN)

Residual neural network (RNN)

Highway network

Mamba

Autoencoder

Variational autoencoder (VAE)

Generative adversarial network (GAN)

Graph neural network (GNN)

Category