Representation Learning

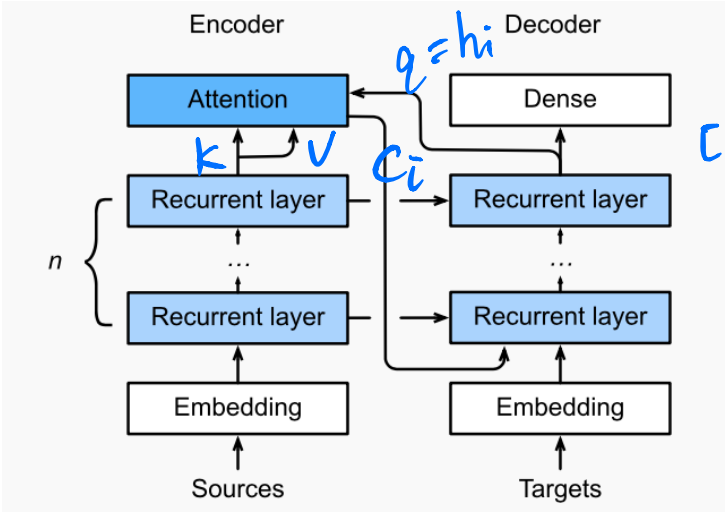
He He

New York University

November 17, 2020

Attention in encoder-decoder models

Loose ends from last lecture:



1. Concatenate current attention output with the decoder hidden state: $y_i = f(y_{i-1}, [h_i; c_i])$
2. Concatenate previous attention output with the decoder input: $y_i = f(y_{i-1}, [y_{i-1}; c_{i-1}])$

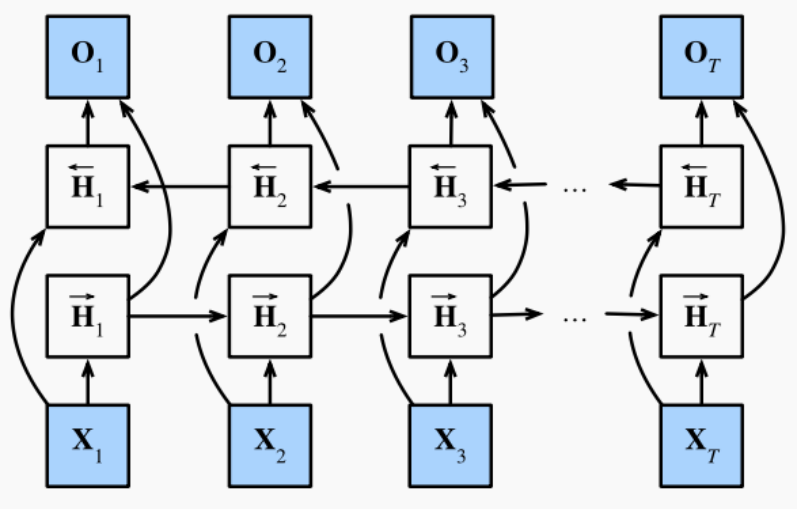
Table of Contents

1. Notable neural architectures in NLP

2. Pre-trained models

3. Autoencoders and VAEs

BiLSTM



- Classification: $p(y \mid x) = \text{softmax}(\text{linear}(\text{pooling}(o_1, \ldots, o_T)))$
- Sequence labeling: $p(y_t \mid x) = \text{softmax}(\text{linear}(o_t))$
- Sequence generation: decoder + attention

Tasks with two inputs

Natural language inference

Premise: 8 million in relief in the form of emergency housing.

Hypothesis: The 8 million dollars for emergency housing was still not enough to solve the problem.

Label: neutral

Reading comprehension

> Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California.

Question: Which team won Super Bowl 50?

Answer: Denver Broncos

Encode two inputs

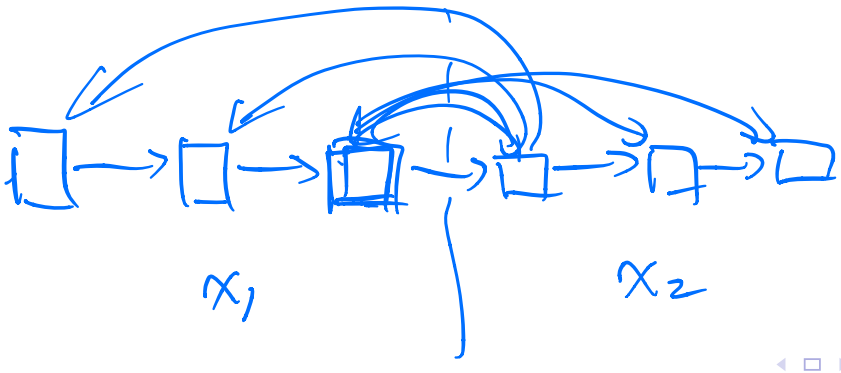
Goal: $\mathcal{X} \times \mathcal{X} \to \mathcal{Y}$

Simple combination:

- Encode x_1 and x_2 in $\mathbb{R}^d$ separately
- Aggregate the two embeddings, e.g. $\text{MLP}(\text{pooling}(\text{enc}(x_1), \text{enc}(x_2)))$
- Pooling: concatenation, elementwise max, elementwise product etc.

Finer-grained interaction between the two inputs:

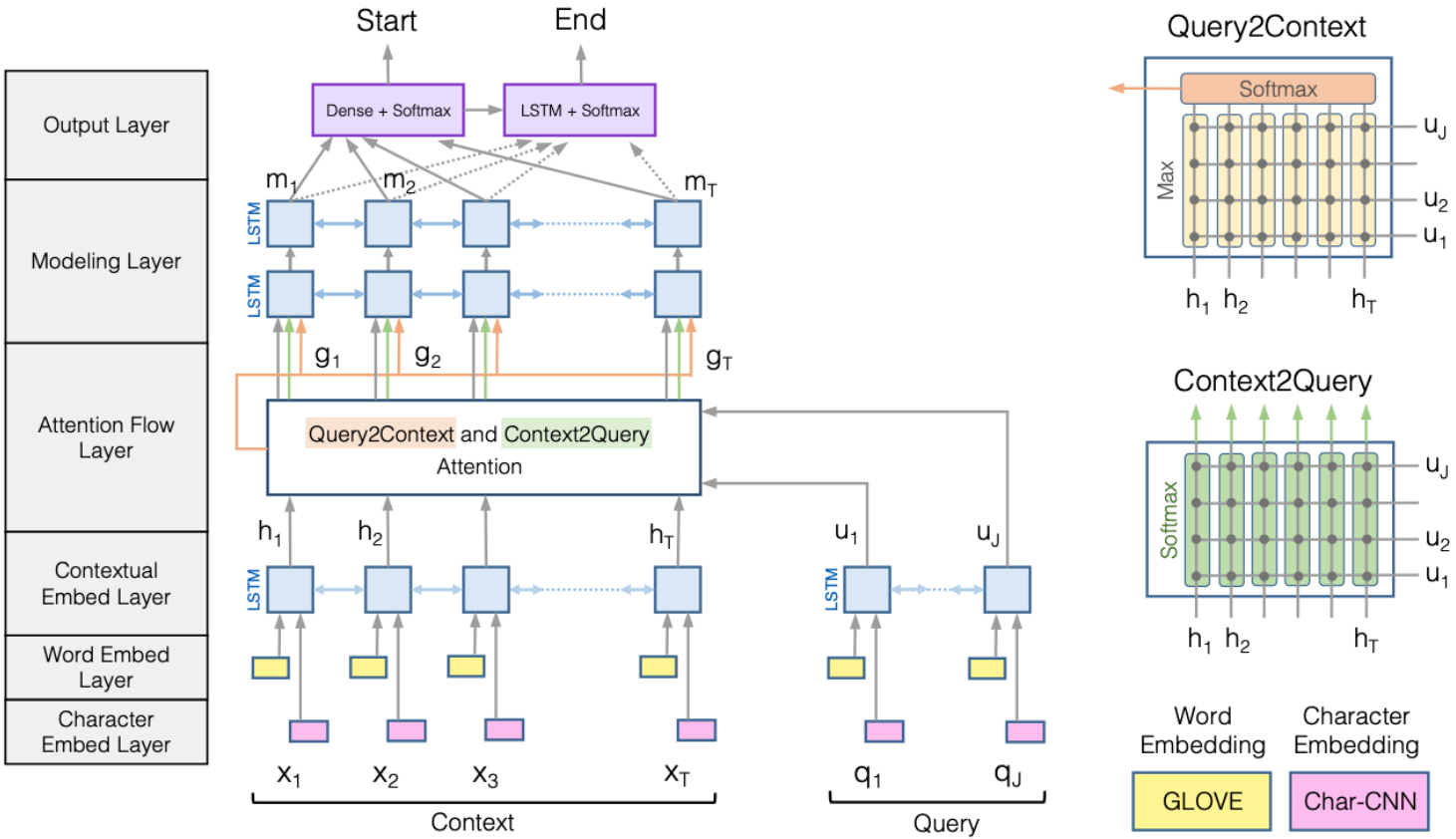
- Can we use something similar to the attention mechanism in seq2seq?



BiDAF

Bi-Directional Attention Flow for Machine Comprehension [Seo+ 2017]

Key idea: representation of x_1 depends on x_2 and vice versa



Attention is all you need?

Word embedding: representation of word meaning

Recurrent neural networks: incorporate variable-length context

Attention mechanism: better modeling of long-range context

Multi-layer biLSTM with various attentions was the go-to architecture for most NLP tasks.

But: RNNs are sequential and hard to scale

We want deeper models trained with larger data.

Can we get rid of the recurrence and use only attention to access context?

Transformer overview

Attention is all you need. [Vaswani+ 2017]

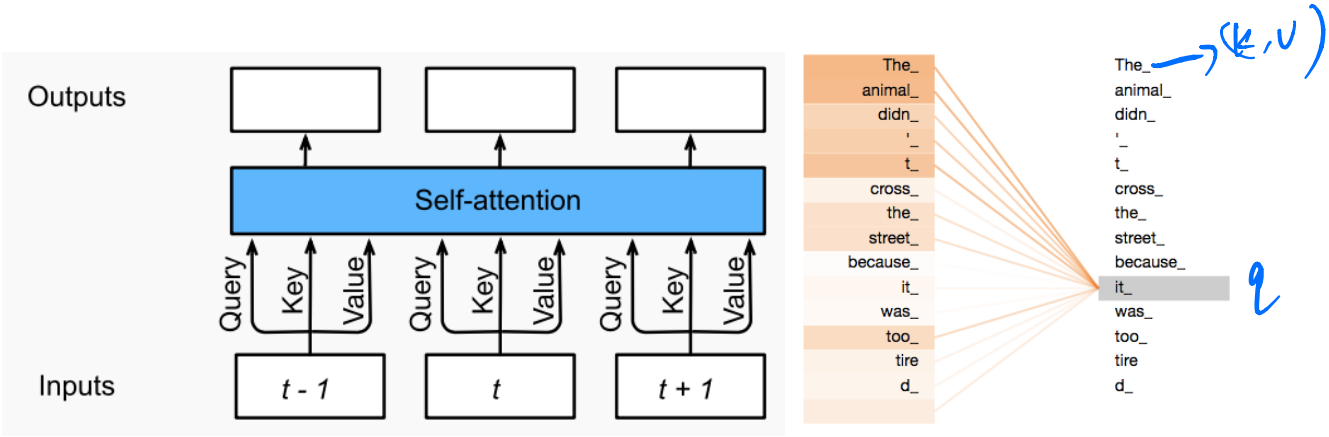
Replaces recurrence with:

- Self-attention: an attention memory of the input sequence for each token in the input sequence (parallelizable)
- Position embedding: models sequential information (doesn't lose word order)

Other key components:

- Multi-head attention
- Residual connection and layer norm: improves optimization of deep models

Self-attention



- Same attention memory (K, V) for all words (parallelizable)
- Each word (as a query) interacts with all words in the input

Matrix representation

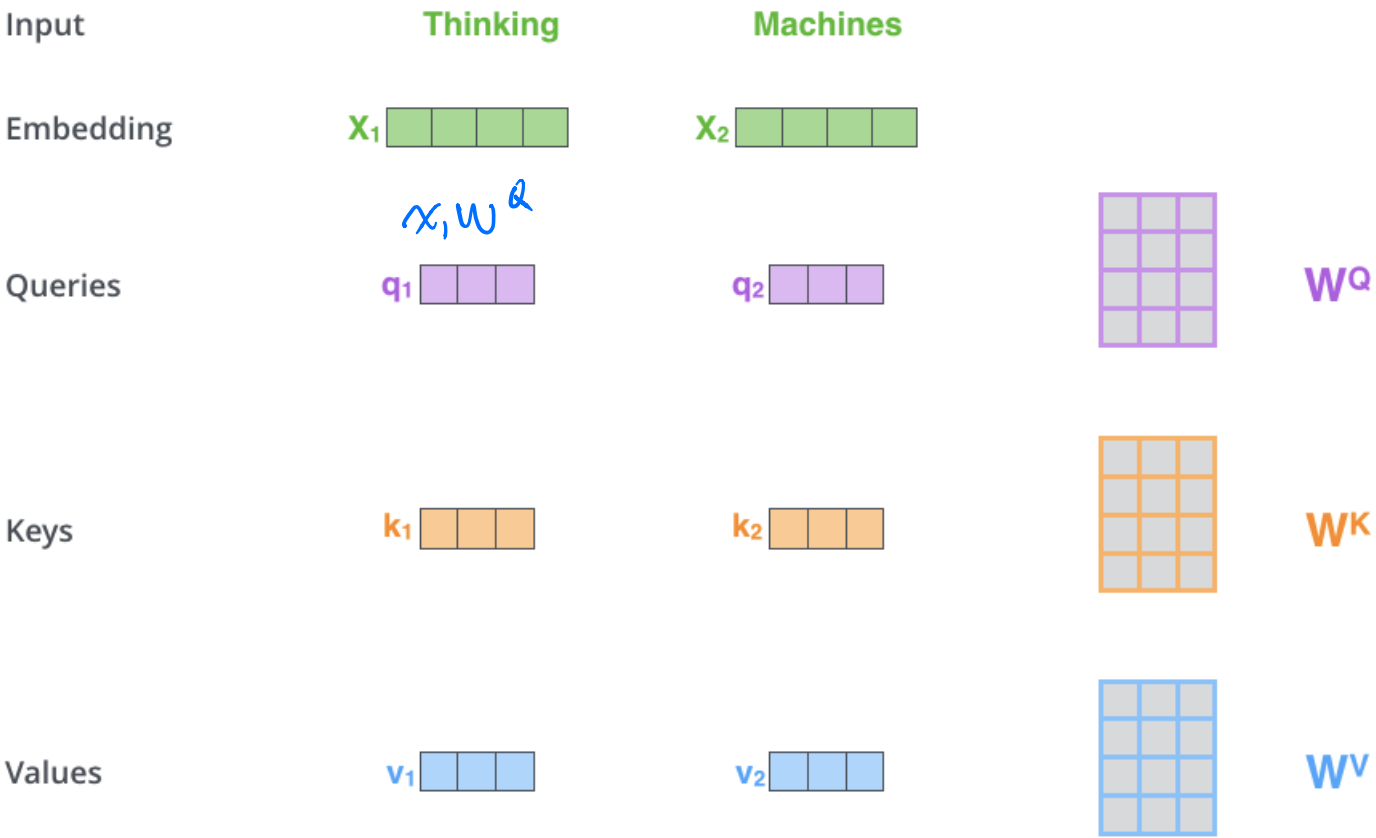
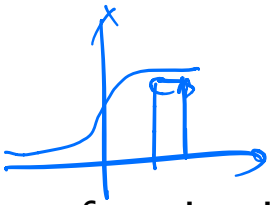


Figure: From "The Illustrated Transformer"

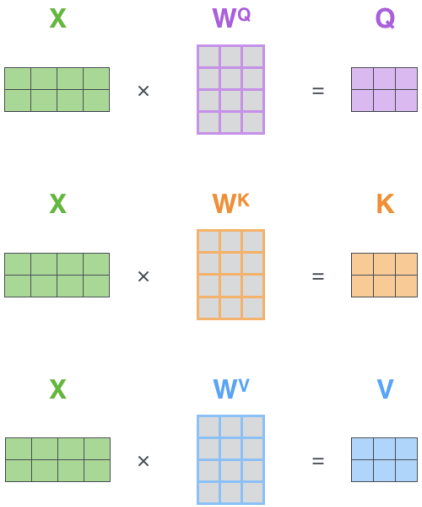
Scaled dot-product attention

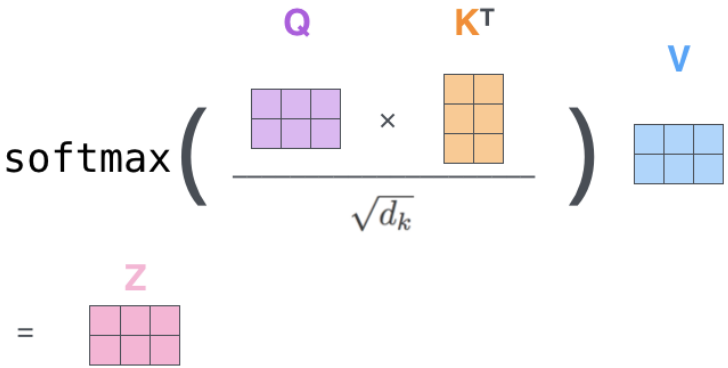
Scaled dot-product attention

$$\alpha(q, k) = q \cdot k / \sqrt{d_k}$$

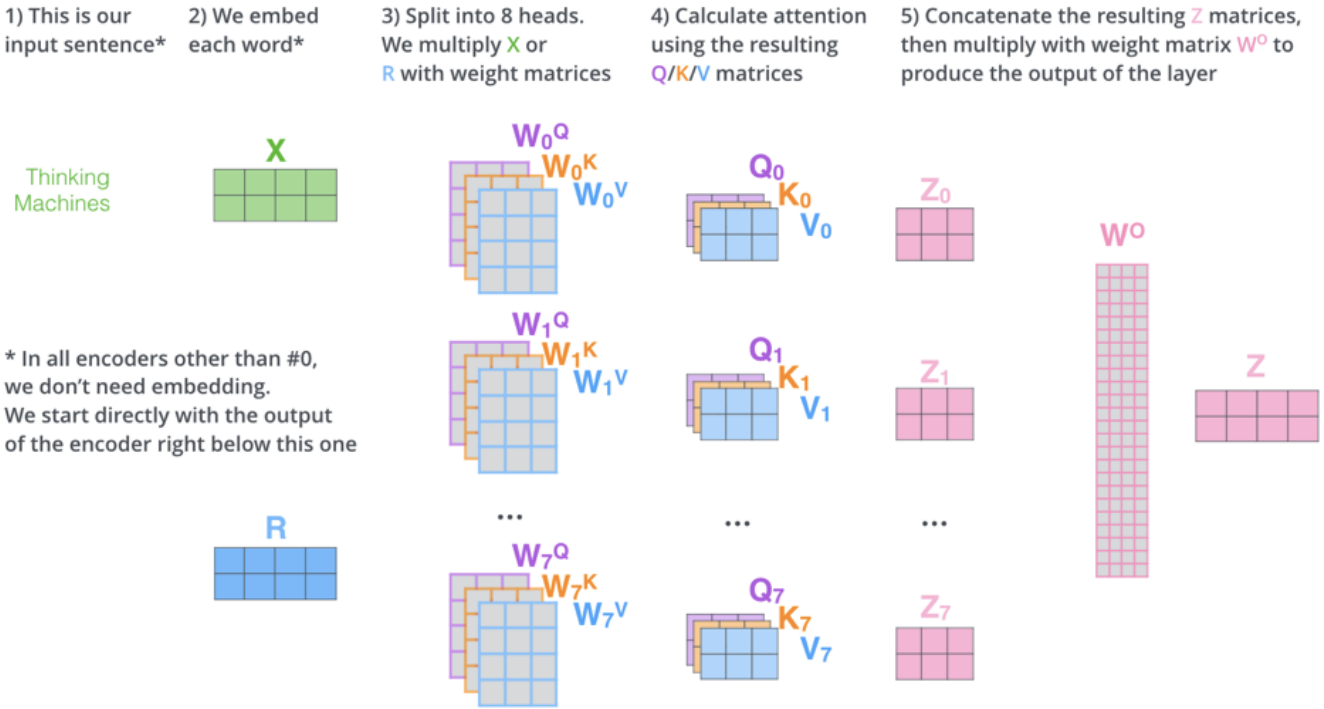


- $\sqrt{d_k}$: dimension of the key vector
- Avoids large attention weights that push the softmax function into regions of small gradients





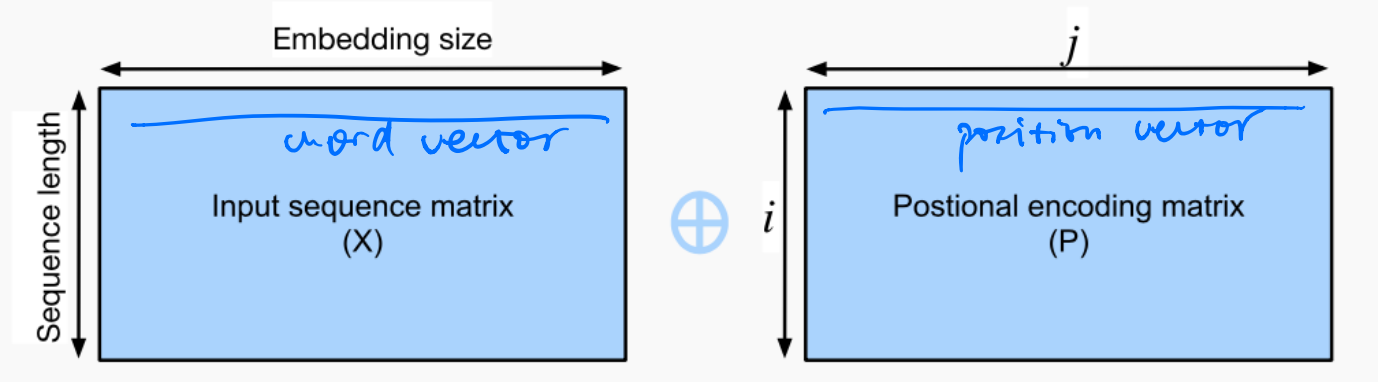
Multi-head attention



Position embedding

Motivation: model word order in the input sequence

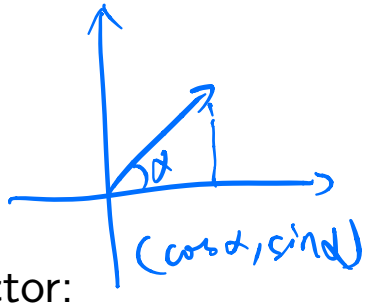
Solution: add a position embedding to each word



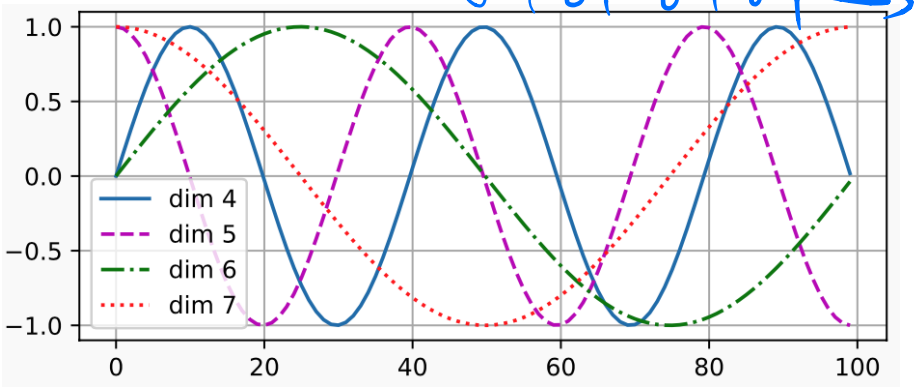
Position embedding:

- Encode absolute and relative positions of a word
- (Same dimension as word embeddings)
- Learned or deterministic

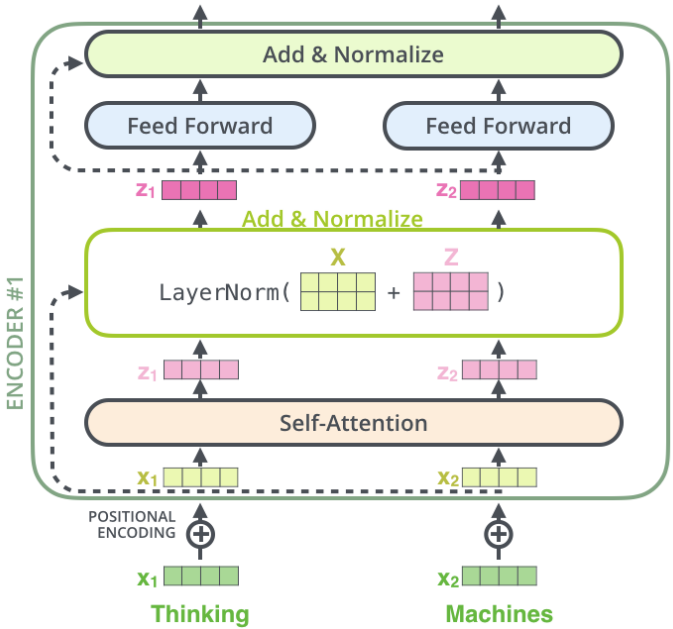
Sinusoidal position embedding



- Same dimension as word embeddings
- Two components $(2i, 2i+1)$ represent a rotation vector: $\sin(\omega_{2i} t), \cos(\omega_{2i} t)$ (t is the position)
- Multiple rotation vectors with different angular velocities: ω_{2i}
- Analogous to binary encoding

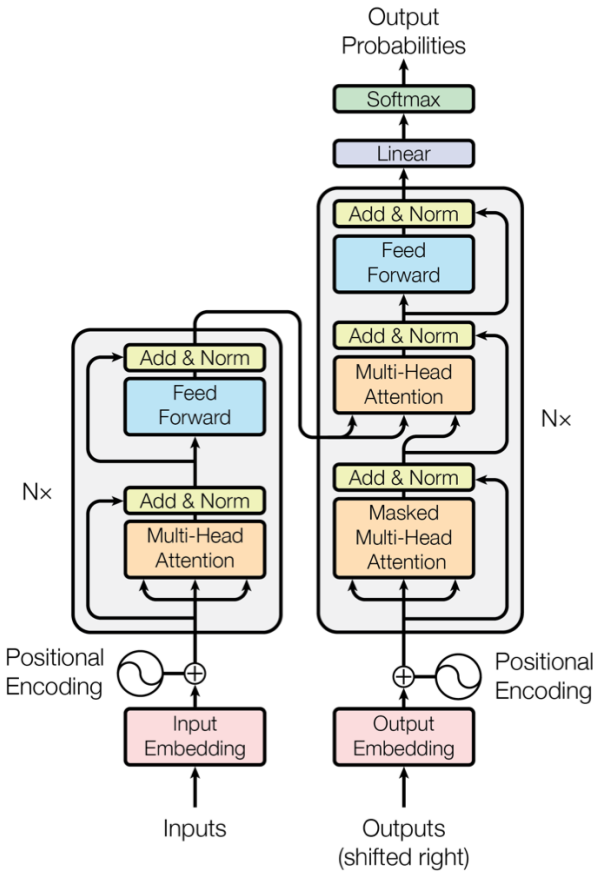


Residual connection and layer normalization



- Residual connection: add input to the output of each layer
- Layer normalization: normalize (zero mean, unit variance) over all features for each sample in the batch
- Position-wise feed-forward networks: same mapping for all positions

Transformer as an encoder-decoder model



- Stacked transformer block
- Decoder attention:
  - Autoregressive generation
  - Self-attention over prefix
  - Encoder-decoder attention over inputs
- Output at each position:

$$p(y_t \mid x, y_{1:t-1})$$

reference

- MLE training

Impact in NLP

- Initially designed for sequential data and obtained SOTA results on MT
- Replaced recurrent models (e.g. LSTM) on many tasks
- Enabled large-scale training which led to pre-trained models such as BERT and GPT-2

Minor limitation: fixed length input. (see Longformer, Performer etc.)

Table of Contents

1. Notable neural architectures in NLP

2. Pre-trained models

3. Autoencoders and VAEs

Representation learning

What are good representations?

Allow for easy extraction of information useful to a learning task

Example:

negative the food is good but doesn't worth an hour wait

Simple features (e.g. BoW) require complex models.
Good features only need simple (e.g. linear) models.

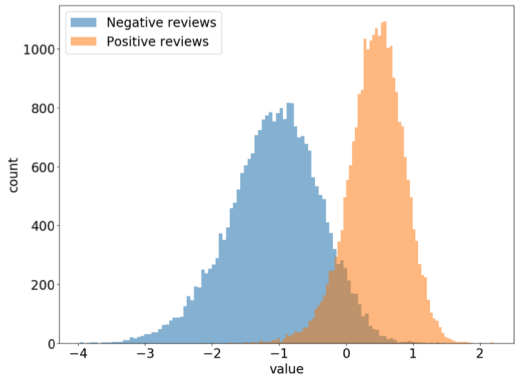


Figure: Sentiment neuron [Radford+ 2017]

Representation learning

Applications of good representations:

- Learning with small data: fine-tuning on learned representations
- Multi-task and transfer learning: shared factors across tasks
- Domain adaptation: more abstract features are less sensitive to domain-specific variations
- Clustering

What should be the objective of representation learning (unsupervised)?

- Self-supervised learning: obtain representations through generative modeling
- Auto-encoders: directy learn the mapping from input to a (latent) representation

Self-supervised learning

Key idea: predict parts of the input from the other parts

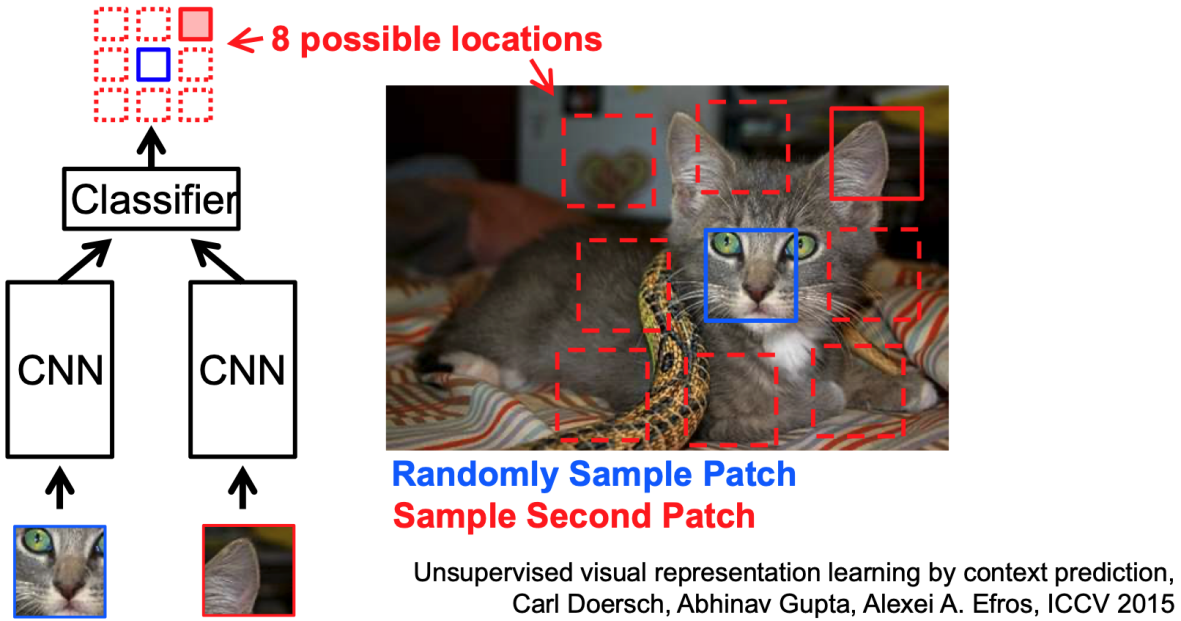


Figure: Slide from Andrew Zisserman

- Other supervision signals: color, rotation etc.
- Video: predict future frames from past frames

Representation learning in NLP

Word embeddings

- CBOW, Skip-gram, GloVe, fastText etc.
- Used as the input layer and aggregated to form sequence representations

Sentence embeddings

- Skip-thought, InferSent, universal sentence encoder etc.
- Challenge: sentence-level supervision

Can we learn something in between?

Word embedding with contextual information

Transfering knowledge from neural LM

Key idea: use representation from a generative model (i.e. an LM)

- Representation (e.g. hidden state at each word) is context-sensitive
- Contains relevant contextual information for predicting the next word

Early work:

- Fine-tune a recurrent LM for downstream tasks [Dai+ 2015, Howard+ 2018]
- Use word embedding from a pre-trained LM in addition to standard word embedding [Peters+ 2017]
- Promising results on a smaller scale

Embeddings from language models (ELMo) [Peters+ 2018]

- Use word embeddings from a bi-directional LM
- Success on multiple NLP tasks

ELMo pretraining

Forward/backward language models:

- $p_{\text{fwd}}(x) = \prod_{t=1}^{T} p(x_t \mid \underbrace{x_{1:t-1}}_{\text{past}}; \theta_{\text{fwd}})$
- $p_{\text{bwd}}(x) = \prod_{t=T}^{1} p(x_t \mid \underbrace{x_{t+1:T}}_{\text{future}}; \theta_{\text{bwd}})$
- Each LM is a two layer LSTM, with shared input embedding layer and softmax layer

Subword representation:

- First layer word embedding is from character convolutions

Data: one-billion word benchmark (monolingual data from WMT)

ELMo embeddings

Contextual embeddings capture word senses.

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

Figure: From [Peters+ 2018].

ELMo Fine-tuning

Obtain contextual word embeddings from each layer $j \in 0, \ldots, L$ of biLM:

$$\text{Embed}(x_t, j) = \begin{cases} [\overrightarrow{h}_{t,j}; \overleftarrow{h}_{t,j}] & \text{for } j > 0 \\ \text{CharEmbed}(x_t) & \text{for } j = 0 \end{cases}$$

Task-specific combination of embeddings:

$$\text{Embed}(x_t) = \gamma \sum_{j=0}^{L} w_j \text{Embed}(x_t, j)$$

Fix biLM and use the contextual word embeddings as input to task-specific models. (Can also add to the output layer.)

Regularization is important: L_2 or dropout.

ELMo results

Improvement on a wide range on NLP tasks:

- reading comprehension (SQuAD)
- entailment/natural language inference (SNLI)
- semantic role labeling (SRL)
- coreference resolution (Coref)
- named entity recognition (NER)
- sentiment analysis (SST-5)

TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Takeaways

- Main idea: use biLM for representaiton learning
- Outputs from all layers are useful
  - Lower layer is better for syntactic tasks, e.g. POS tagging, parsing
  - Hight layer is better for semantic tasks, e.g. question answering, NLI
  - Some fine-tuning of the pre-trained model is needed.
- Large-scale training is important

Next, pre-trained transformer models.

Transformer models

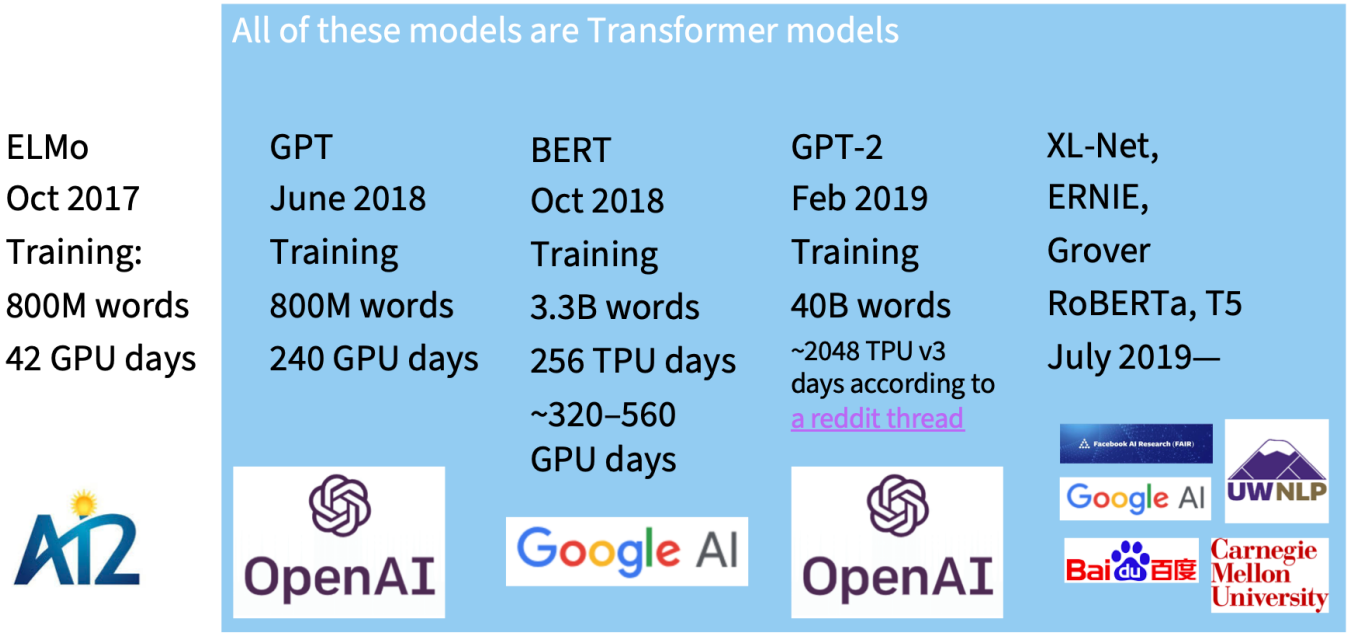


Figure: Slide from Chris Manning

Bidirectional Encoder Representations from Transformers (BERT)

Pre-training:

1. Masked LM:

$$\mathbb{E}_{x \sim \mathcal{D}, i \sim p_{\text{mask}}} \log p(x_i \mid x_{-i}; \theta)$$

(not a LM) $\prod_i p(x_i \mid x_{-i}) \neq p(x)$

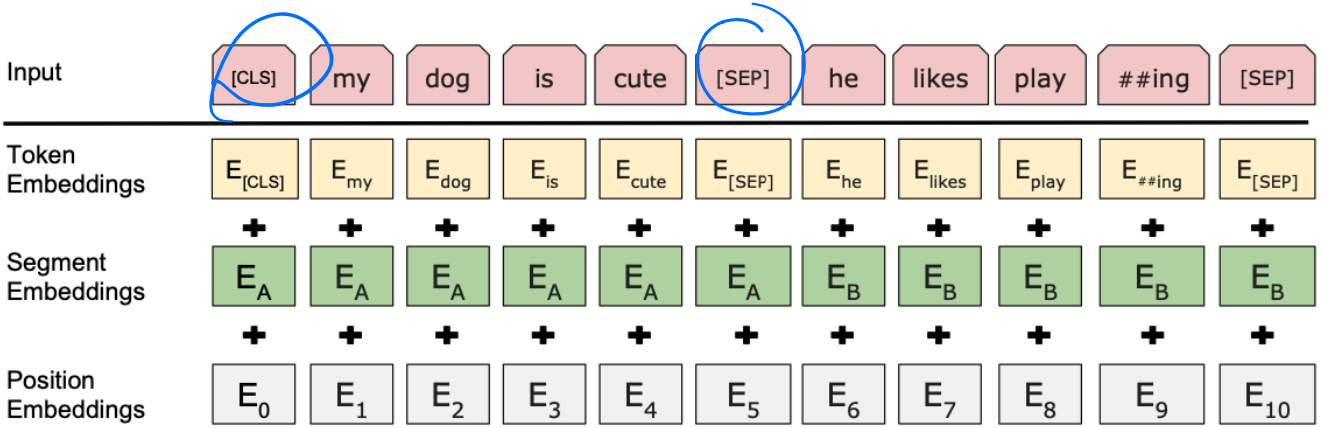
   - x_{-i}: noised version of x where x_i is replaced by `[MASK]`, a random token, or the original token
   - $p(x_i \mid x_{-i}; \theta) = \text{Transformer}(x_{-i}, i)$

2. Next sentence prediction:

$$\mathbb{E}_{x^1 \sim \mathcal{D}, x^2 \sim p_{\text{next}}} \log p(y \mid x^1, x^2)$$

   - y: whether x^2 follows x^1
   - Not as useful as masked LM

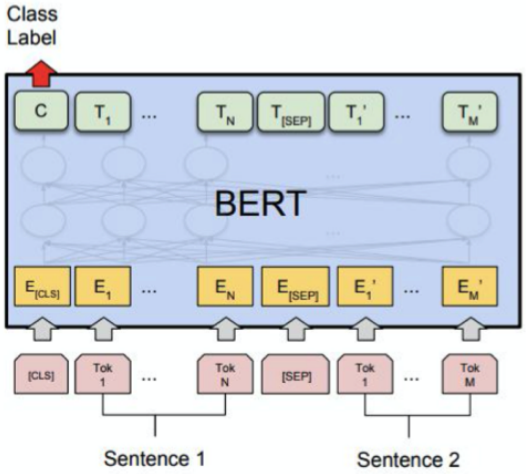
BERT sentence pair encoding



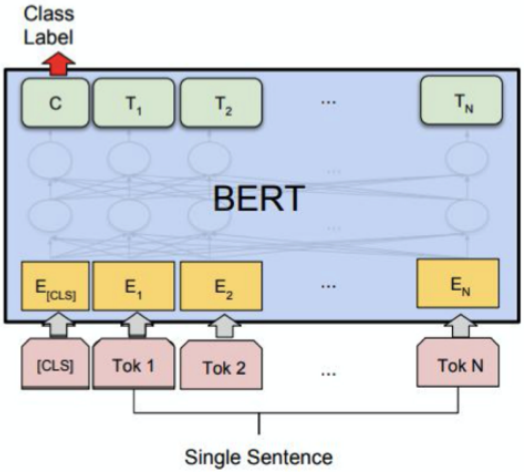
- [CLS]: first token of all sequences; used for next sentence prediction
- Distinguish two sentences in a pair: [SEP] and segment embedding
- Learned position embedding
- Subword unit: wordpiece (basically byte pair encoding)

BERT fine-tuning

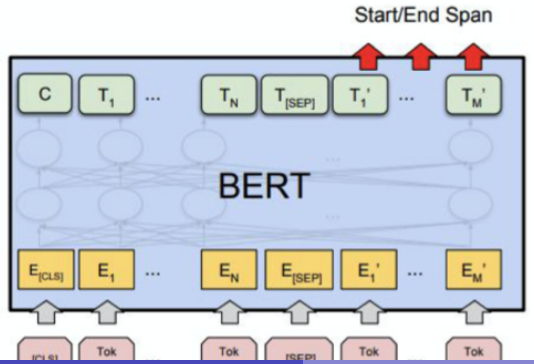
All weights are fine-tuned (with a small learning rate)

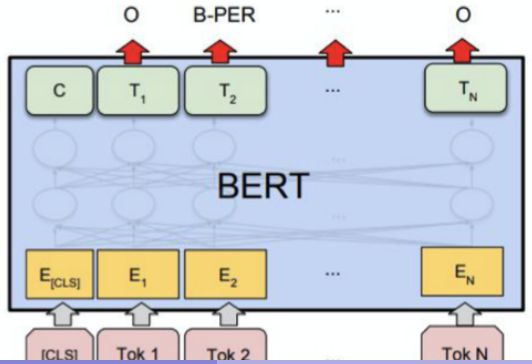


(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA





Recent progress

GLUE: benchmark of natural language understanding tasks

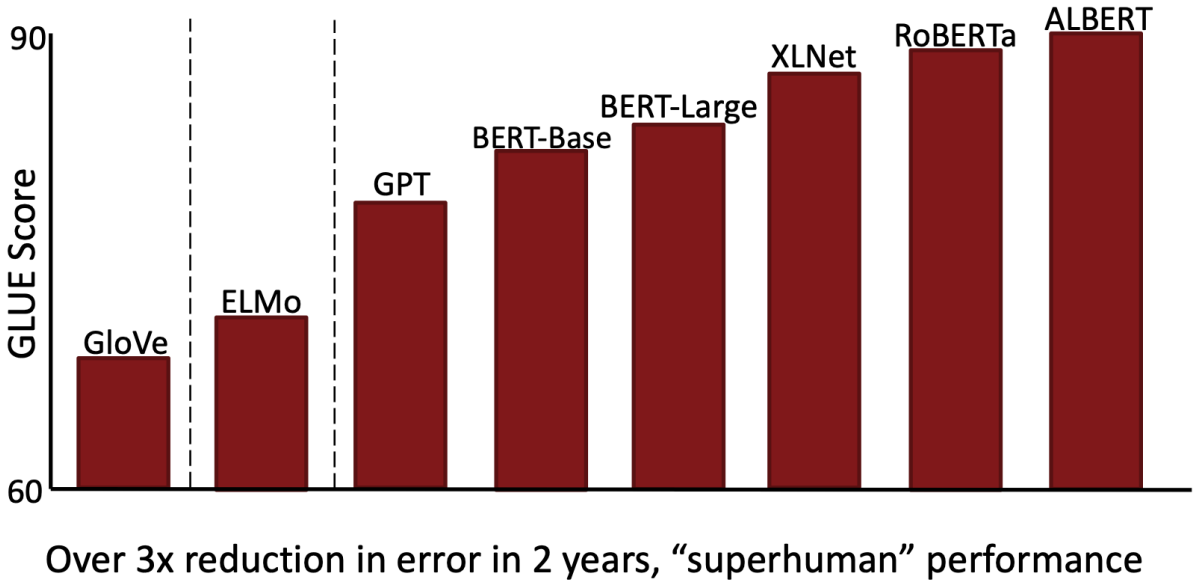


Figure: Slide from Chris Manning

Summary

Off-the-shelf solution for NLP tasks: fine-tune BERT (and friends)

What's next?

- Processing long text
- Efficient training/inference
- Learning with a small amount of data
- Generalize to new test distributions (solve tasks, not datasets)

Table of Contents

1. Notable neural architectures in NLP

2. Pre-trained models

3. Autoencoders and VAEs

Overview

Motivation: Directly learn a (parametric) mapping from the input to the representation

Encoder: dimensionality reduction (data to code)

$$z = \text{enc}_\theta(x)$$

Decoder: reconstruction (code to data)

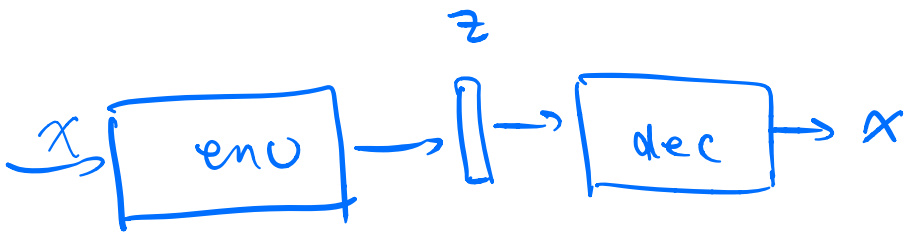
$$x = \text{dec}_\gamma(\text{enc}_\theta(x))$$

Learning: reconstruction loss

$$J(\theta, \gamma) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} L(x, \text{dec}_\gamma(\text{enc}_\theta(x)))$$

L: MSE, cross-entropy etc.

Autoencoder



- Parametrize encoder and decoder by neural networks
- What model can we use if x is text?
- Problem: z could just copy the input x and no meaningful representation is learned!

Denoise autoencoder

Intuition: structural information is needed to recover a corrupted input

Learning: reconstruction clean input from a corrputed version

$$J(\theta, \gamma) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \mathbb{E}_{\tilde{x} \sim p_c(\cdot|x)} \left[L(x, \text{dec}_\gamma(\text{enc}_\theta(\tilde{x}))) \right]$$

- Corruption: delete/insert/substitute words, Gaussian noise for images
- Which model we have seen can be considered as a denoise autoencoder? BERT

Variational autoencoder

Model z as a latent variable:

$$p(x;\theta) = \sum_{z\in\mathcal{Z}} p(x,z;\theta) \quad \left(\int_z p(x,z;\theta)dz \quad \text{if } z \text{ is continuous}\right)$$

Recall the evidence lowerbound (see lecture on EM)

variational dist

$$\begin{aligned}\text{ELBO} &= \log p(x;\theta) - \text{KL}\left(q(z)\|p(z\mid x;\theta)\right)\\ &= \sum_{z\in\mathcal{Z}} q(z)\log\frac{p(x,z;\theta)}{q(z)}\end{aligned}$$

Concept check: What is $q(z)$ for EM? $p(z|x;\theta)$

In general, we can learn $q(z)$ for each sample:

$$\max_\theta \sum_{x\in\mathcal{D}} \max_\gamma \sum_{z\in\mathcal{Z}} q(z;\gamma)\log\frac{p(x,z;\theta)}{q(z;\gamma)}$$

Variational autoencoder

Objective:

$$\max_{\theta} \sum_{x \in \mathcal{D}} \max_{\gamma} \sum_{z \in \mathcal{Z}} q(z; \gamma) \log \frac{p(x, z; \theta)}{q(z; \gamma)}$$

EM-style learning:

1. Solve γ for each sample (using SGD):

$$\gamma_x^* \leftarrow \max_{\gamma} \text{ELBO}(x; \theta^{\text{old}}, \gamma)$$

2. One-step update θ^{old}:

$$\theta^{\text{new}} \leftarrow \theta^{\text{old}} + \nabla_{\theta} \sum_{x \in \mathcal{D}} \text{ELBO}(x; \theta, \gamma_x^*)$$

Problems:

- Step 1 is expensive!
- How to compute the gradient?

Amortized variational inference

Problem: finding the optimal variation distribution for each sample x is expensive

$$\gamma_x^* \leftarrow \max_{\gamma} \text{ELBO}(x; \theta^{\text{old}}, \gamma)$$

Idea: learn the mapping $f_\phi : x \to \gamma_x^*$

Note that $f_\phi(x)$ specifies the distribution $q(z; \gamma_x)$. Let's use $q(z \mid x; \phi)$ to denote $q_\gamma(z)$.

Estimate ϕ (same for all x)

$$\max_{\phi} \sum_{x \in \mathcal{D}} \text{ELBO}(x; \theta, \phi)$$

$$= \max_{\phi} \sum_{x \in \mathcal{D}} \mathbb{E}_{q(z|x;\phi)} \left[\log \frac{p(x, z, \theta)}{q(z \mid x; \phi)} \right]$$

$p(x)$

$p(x|z;\theta)p(z;\theta)$

$KL(q(z|x) \| p(z))$

Update ϕ and θ iteratively for each mini-batch.

Gradient computation

Need to compute $\nabla_\phi \text{ELBO}(x; \phi, \theta)$ and $\nabla_\theta \text{ELBO}(x; \phi, \theta)$

Monte Carlo sampling (REINFORCE):

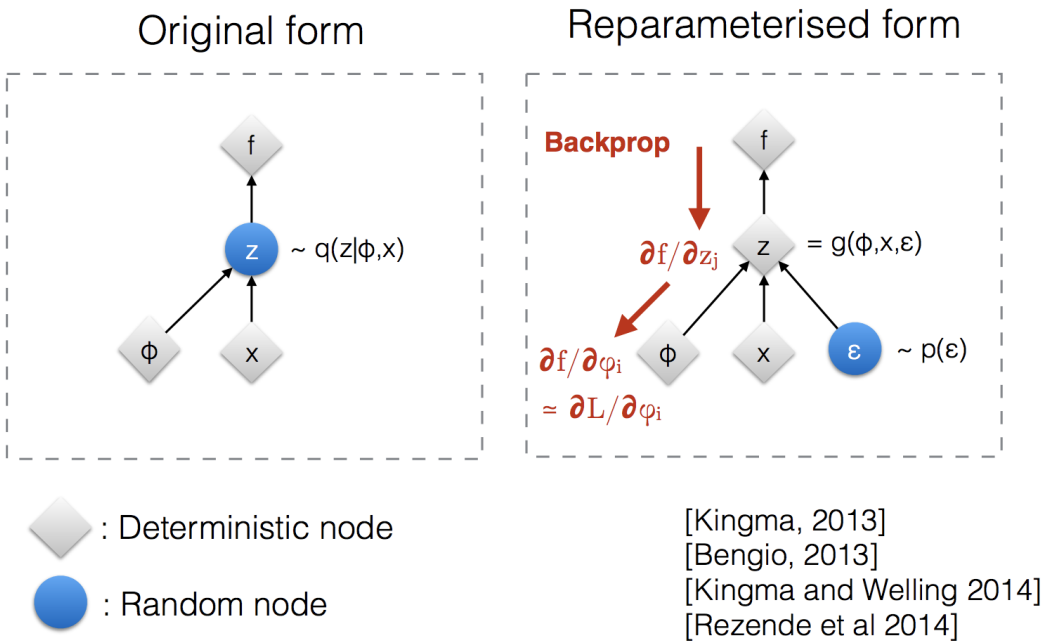
$$\nabla_\phi \mathbb{E}_{q(z|x;\phi)} \left[\log \frac{p(x, z; \theta)}{q(z \mid x; \phi)} \right] = \mathbb{E}_{q(z|x;\phi)} \left[\nabla_\phi q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)} \right]$$

$$\approx \frac{1}{n} \sum_{i=1}^{n} \left[\nabla_\phi q(z^{(i)} \mid x; \phi) \log \frac{p(x, z^{(i)}; \theta)}{q(z^{(i)} \mid x; \phi)} \right]$$

where $z^{(i)} \sim q(z \mid x; \phi)$.

Problem: gradient estimator has high variance.

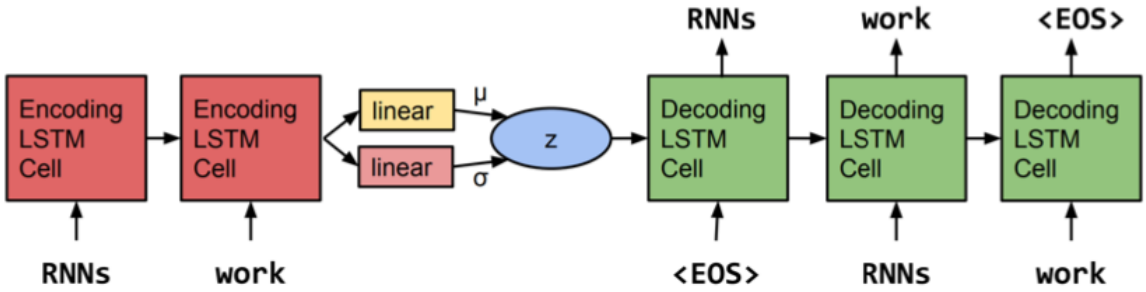
Reparametrization trick



[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

Original distribution: $z^{(i)} \sim q(z \mid x; \phi)$
Auxiliary independent random variable: $\epsilon \sim p(\epsilon)$
Reparametrization trick: $z = \mathcal{T}(\epsilon, x; \phi)$

Parametrization by neural networks



Joint distribution: $p(x, z; \theta) = p(x \mid z; \theta)p(z; \theta)$

Prior: $p(z; \theta) = \mathcal{N}(z \mid 0, I)$

Likelihood: $p(x \mid z; \theta) = \text{dec}_\theta(z)$

Variational distribution: $q(z \mid x; \phi) = \text{enc}_\phi(x) = \mu_\phi(x), \sigma^2_\phi(x)$

Reparametrization:

$$\epsilon \sim \mathcal{N}(0, I)$$
$$z = \mu_\phi(x) + \sigma^2_\phi(x) \cdot \epsilon$$

Summary

- Autoencoders: directly learn a low-dimensional representation through reconstruction
- VAEs: impose structure on the latent representation
  - In addition to representation learning, also used for latent variable models in NLP
  - Issues: posterior collapse, evaluation