DSC250: Advanced Data Mining

Topic Models / Text Embedding

Zhiting Hu

Lecture 9, Feb 4, 2025

UC San Diego
HALICIOĞLU DATA SCIENCE INSTITUTE

Recap: Expectation Maximization (EM)

- Supervised MLE is easy: $\max_\theta \; \ell_c(\theta; \boldsymbol{x}, \boldsymbol{z}) = \log p(\boldsymbol{x}, \boldsymbol{z}|\theta)$
  - Observe both $\boldsymbol{x}$ and $\boldsymbol{z}$
- Unsupervised MLE is hard: $\max_\theta \; \ell(\theta; \boldsymbol{x}) = \log p(\boldsymbol{x}|\theta) = \log \sum_z p(\boldsymbol{x}, \boldsymbol{z}|\theta)$
  - Observe only $\boldsymbol{x}$
- EM, intuitively:

E-step: $q(\boldsymbol{z}|\boldsymbol{x}) = p(\boldsymbol{z}|\boldsymbol{x}, \theta)$

We don't actually observe q, let's estimate it

M-step: $\max_\theta \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\, \log p(\boldsymbol{x}, \boldsymbol{z}|\theta) \,]$

Let's "pretend" we also observe $\boldsymbol{z}$ (its distribution)

Recap: Expectation Maximization (EM)

- Supervised MLE is easy:
  - Observe both $\boldsymbol{x}$ and $\boldsymbol{z}$

$$\max_{\theta} \ \ell_c(\theta; \boldsymbol{x}, \boldsymbol{z}) = \log p(\boldsymbol{x}, \boldsymbol{z} | \theta)$$

- Unsupervised MLE is hard:
  - Observe only $\boldsymbol{x}$

$$\max_{\theta} \ \ell(\theta; \boldsymbol{x}) = \log p(\boldsymbol{x} | \theta) = \log \sum_{z} p(\boldsymbol{x}, \boldsymbol{z} | \theta)$$

- EM, intuitively:

E-step: $q^{t+1}(\boldsymbol{z}|\boldsymbol{x}) = p(\boldsymbol{z}|\boldsymbol{x}, \theta^t)$

We don't actually observe q, let's estimate it

M-step: $\max_{\theta} \mathbb{E}_{q^{t+1}(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}, \boldsymbol{z}|\theta)]$

Let's "pretend" we also observe $\boldsymbol{z}$ (its distribution)

This is an iterative process

Recap: Expectation Maximization (EM)

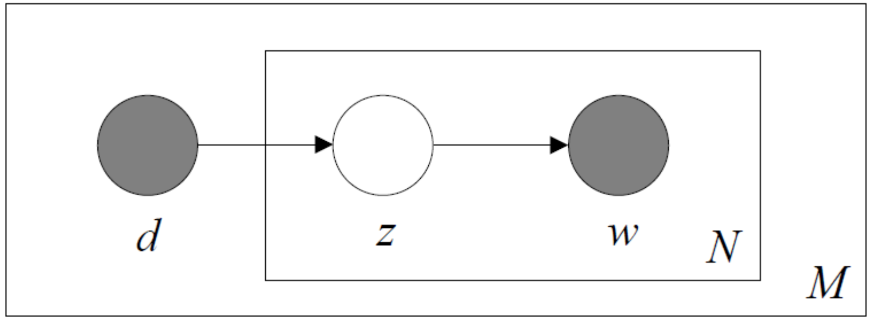
- The EM algorithm is coordinate-decent on $F(q, \theta)$
  - E-step: $q^{t+1} = \arg\min_q F\left(q, \theta^t\right) = p(\boldsymbol{z}|\boldsymbol{x}, \theta^t)$
    - the posterior distribution over the latent variables given the data and the current parameters
  - M-step: $\theta^{t+1} = \arg\min_\theta F\left(q^{t+1}, \theta^t\right) = \text{argmax}_\theta \sum_z q^{t+1}(\boldsymbol{z}|\boldsymbol{x}) \log p(\boldsymbol{x}, \boldsymbol{z}|\theta)$

ELBO

$$\ell(\theta; \boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z}|\theta)}{q(\boldsymbol{z}|\boldsymbol{x})}\right] + \underbrace{\mathrm{KL}\big(q(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z}|\boldsymbol{x}, \theta)\big)}_{\geq 0}$$

$$= -F(q, \theta) + \mathrm{KL}\big(q(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z}|\boldsymbol{x}, \theta)\big)$$

Recap: Learning pLSA with EM



- **Likelihood function of a word w:**

$$p(w|d,\theta,\beta) = \sum_k p(w, z = k|d,\theta,\beta)$$

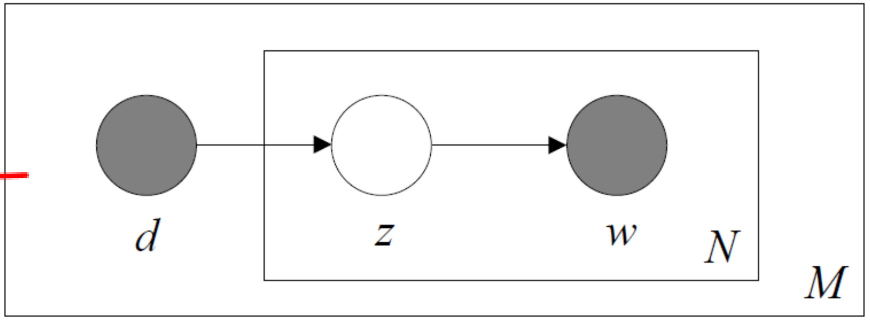
$$= \sum_k p(w|z = k, d,\ ,\beta)p(z = k|d,\theta,\ ) = \sum_k \beta_{kw}\theta_{dk}$$

- **Learning by maximizing the log likelihood:**

$$\max_{\beta,\theta} \log \sum_k \beta_{kw}\theta_{dk}$$

Recap: Learning pLSA with EM

Bayes rule



θ^t, β^t

- E-step:

$$q(z|w,d) = p(z|w,d,\theta^t,\beta^t) = \frac{p(z,w,d,\theta^t,\beta^t)}{\sum_{z'} p(w,d,\theta^t,\beta^t)}$$

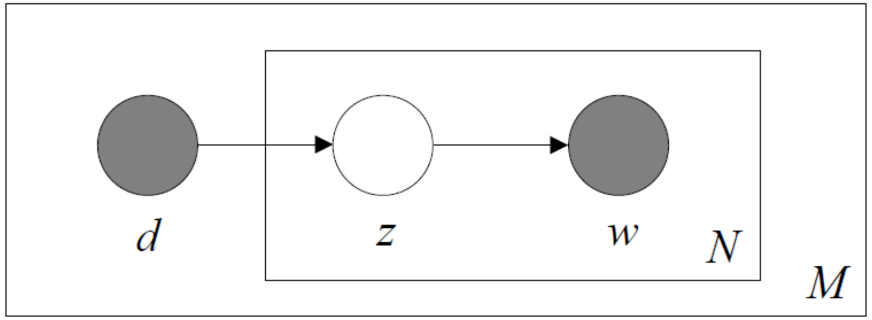
$$= \frac{\beta^t_{zw}\cdot\theta^t_{dz}}{\sum_{z'}\beta^t_{z'w}\cdot\theta^t_{dz'}}$$

- M-step:

$$\beta^t_{zw}\theta^t_{dz}$$

$$\max_{\beta,\theta} E_{q(z|w,d)}\left[\log p(z,w,d,\theta^t,\beta^t)\right]$$

Recap: Learning pLSA with EM



- E-step:

$$p(z|w,d,\theta^t,\beta^t) = \frac{p(w|z,d,\beta^t)p(z|d,\theta^t)}{\sum_{z'} p(w|z',d,\beta^t)p(z'|d,\theta^t)} = \frac{\beta^t_{zw}\theta^t_{dz}}{\sum_{z'} \beta^t_{z'w}\theta^t_{dz'}}$$

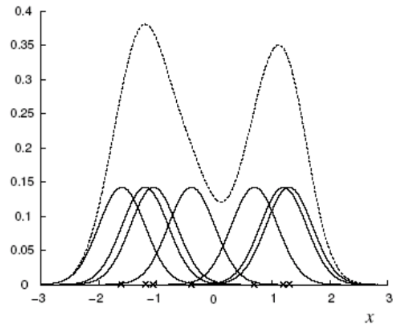
- M-step:

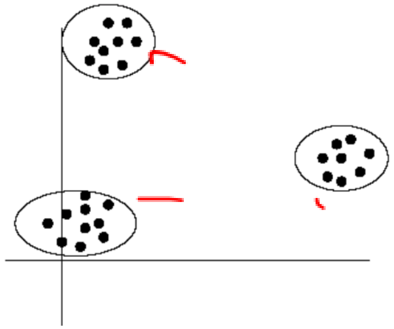
Another Example: Gaussian Mixture Models (GMMs)

- Consider a mixture of K Gaussian components:

$$p(x_n|\mu,\Sigma) = \sum_k \pi_k N(x,|\,\mu_k,\Sigma_k)$$

mixture proportion　　mixture component



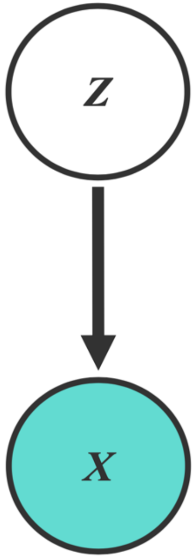


K = 3

$z \in \{0, 1, 2\}$

- This model can be used for unsupervised clustering.
  - This model (fit by AutoClass) has been used to discover new kinds of stars in astronomical data, etc.

Example: Gaussian Mixture Models (GMMs)



- Consider a mixture of K Gaussian components:
  - Z is a latent class indicator vector:

$$p(z_n) = \mathrm{multi}(z_n : \pi) = \prod_k (\pi_k)^{z_n^k}$$

  - X is a conditional Gaussian variable with a class-specific mean/covariance

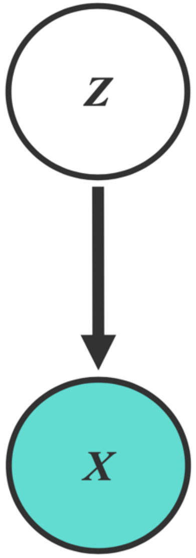
$$p(x_n \mid z_n^k = 1, \mu, \Sigma) = \frac{1}{(2\pi)^{m/2}|\Sigma_k|^{1/2}} \exp\left\{-\tfrac{1}{2}(x_n - \mu_k)^T \Sigma_k^{-1} (x_n - \mu_k)\right\} = \mathcal{N}(x_n \mid \mu_{z_n}, \Sigma_{z_n})$$

  - The likelihood of a sample:

$$P(x \mid \pi, \mu, \Sigma) = \sum_z P(z \mid \pi) P(x \mid z, \mu, \Sigma)$$

$$\log P(x \mid \pi, \mu, \Sigma) = \log \sum_z \cdots$$

Example: Gaussian Mixture Models (GMMs)



- Consider a mixture of K Gaussian components:
  - Z is a latent class indicator vector:

$$p(z_n) = \mathrm{multi}(z_n : \pi) = \prod_k (\pi_k)^{z_n^k}$$

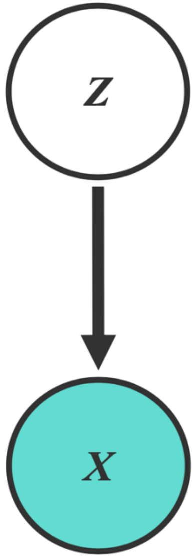
  - X is a conditional Gaussian variable with a class-specific mean/covariance

$$p(x_n \mid z_n^k = 1, \mu, \Sigma) = \frac{1}{(2\pi)^{m/2} |\Sigma_k|^{1/2}} \exp\left\{-\tfrac{1}{2}(x_n - \mu_k)^T \Sigma_k^{-1} (x_n - \mu_k)\right\}$$

  - The likelihood of a sample:

$$\begin{aligned} p(x_n | \mu, \Sigma) &= \sum_k p(z^k = 1 \mid \pi)\, p(x, \mid z^k = 1, \mu, \Sigma) \\ &= \sum_{z_n} \prod_k \left((\pi_k)^{z_n^k} N(x_n : \mu_k, \Sigma_k)^{z_n^k} \right) = \sum_k \pi_k N(x, \mid \mu_k, \Sigma_k) \end{aligned}$$

Example: Gaussian Mixture Models (GMMs)



- Consider a mixture of K Gaussian components:
  - Z is a latent class indicator vector:

$$p(z_n) = \mathrm{multi}(z_n : \pi) = \prod_k (\pi_k)^{z_n^k}$$

  - X is a conditional Gaussian variable with a class-specific mean/covariance

$$p(x_n \mid z_n^k = 1, \mu, \Sigma) = \frac{1}{(2\pi)^{m/2} |\Sigma_k|^{1/2}} \exp\left\{ -\tfrac{1}{2} (x_n - \mu_k)^T \Sigma_k^{-1} (x_n - \mu_k) \right\}$$

  - The likelihood of a sample:

Parameters to be learned:

mixture component

mixture proportion

$$p(x_n | \mu, \Sigma) = \sum_k p(z^k = 1 \mid \pi) p(x, \mid z^k = 1, \mu, \Sigma)$$
$$= \sum_{z_n} \prod_k \left((\pi_k)^{z_n^k} N(x_n : \mu_k, \Sigma_k)^{z_n^k} \right) = \sum_k \pi_k N(x, \mid \mu_k, \Sigma_k)$$

Example: Gaussian Mixture Models (GMMs)

- Consider a mixture of K Gaussian components
- E-step: computing the posterior of z_n given the current estimate of the parameters (i.e., π , μ, Σ)

$$q(z \mid x, \mu^{(t)}, \Sigma^{(t)}, \pi^{(t)}) = P(z \mid x, \mu^{(t)}, \Sigma^{(t)}, \pi^{(t)})$$

$$= \frac{P(z, x, \mu^{(t)}, \Sigma^{(t)}, \pi^{(t)})}{\sum_{z'} P(z', x, \mu^{(t)}, \Sigma^{(t)}, \pi^{(t)})}$$

$$P(z \mid \pi \ldots) \, P(x \mid z, \ldots)$$

$$\pi_z \cdot N(x \mid z \ldots)$$

Example: Gaussian Mixture Models (GMMs)

- Consider a mixture of K Gaussian components
- E-step: computing the posterior of z_n given the current estimate of the parameters (i.e., π , μ, Σ)

$$p(z_n^k = 1 | x, \mu^{(t)}, \Sigma^{(t)}) = \frac{\pi_k^{(t)} N(x_n, | \mu_k^{(t)}, \Sigma_k^{(t)})}{\sum_i \pi_i^{(t)} N(x_n, | \mu_i^{(t)}, \Sigma_i^{(t)})}$$

$p(\underline{z_n^k = 1}, x, \mu^{(t)}, \Sigma^{(t)})$

$p(x, \mu^{(t)}, \Sigma^{(t)})$

Example: Gaussian Mixture Models (GMMs)

- Consider a mixture of K Gaussian components
- E-step: computing the posterior of z_n given the current estimate of the parameters (i.e., π , μ, Σ)

$$p(z_n^k = 1 \mid x, \mu^{(t)}, \Sigma^{(t)}) = \frac{\overbrace{\pi_k^{(t)} N(x_n, \mid \mu_k^{(t)}, \Sigma_k^{(t)})}^{p(z_n^k = 1, x, \mu^{(t)}, \Sigma^{(t)})}}{\underbrace{\sum_i \pi_i^{(t)} N(x_n, \mid \mu_i^{(t)}, \Sigma_i^{(t)})}_{p(x, \mu^{(t)}, \Sigma^{(t)})}}$$

- M-step: the expected complete log likelihood

$$\max_{\pi, \Sigma, \mu} E_{q(z \mid x, \mu^{(t)}, \Sigma^{(t)}, \pi^{(t)})} \left[\log P(z, x, \mu, \Sigma, \pi) \right]$$

Example: Gaussian Mixture Models (GMMs)

- Consider a mixture of K Gaussian components
- E-step: computing the posterior of z_n given the current estimate of the parameters (i.e., π , μ, Σ)

$$p(z_n^k = 1 | x, \mu^{(t)}, \Sigma^{(t)}) = \frac{\pi_k^{(t)} N(x_n, | \mu_k^{(t)}, \Sigma_k^{(t)})}{\sum_i \pi_i^{(t)} N(x_n, | \mu_i^{(t)}, \Sigma_i^{(t)})}$$

$p(z_n^k = 1, x, \mu^{(t)}, \Sigma^{(t)})$

$p(x, \mu^{(t)}, \Sigma^{(t)})$

- M-step: the expected complete log likelihood

$$\mathbb{E}_q \left[\ell_c(\boldsymbol{\theta}; x, z)\right] = \sum_n \mathbb{E}_q \left[\log p\left(z_n \mid \pi\right)\right] + \sum_n \mathbb{E}_q \left[\log p\left(x_n \mid z_n, \mu, \Sigma\right)\right]$$

$$= \sum_n \sum_k \mathbb{E}_q \left[z_n^k\right] \log \pi_k - \frac{1}{2} \sum_n \sum_k \mathbb{E}_q \left[z_n^k\right] \left(\left(x_n - \mu_k\right)^T \Sigma_k^{-1} \left(x_n - \mu_k\right) + \log\left|\Sigma_k\right| + C \right)$$

Example: Gaussian Mixture Models (GMMs)

- M-step: computing the parameters given the current estimate of z_n

$$\pi_k^* = \arg\max \langle l_c(\boldsymbol{\theta}) \rangle, \qquad \Rightarrow \frac{\partial}{\partial \pi_k} \langle l_c(\boldsymbol{\theta}) \rangle = 0, \forall k, \quad \text{s.t.} \sum_{\mathrm{k}} \pi_k = 1$$

$$\Rightarrow \quad \pi_k^* = \sum_n \langle z_n^k \rangle_{q^{(t)}} \Big/ N = \sum_n \tau_n^{k(t)} \Big/ N = \langle n_k \rangle \Big/ N$$

$$\mu_k^* = \arg\max \langle l(\boldsymbol{\theta}) \rangle, \qquad \Rightarrow \quad \mu_k^{(t+1)} = \frac{\sum_n \tau_n^{k(t)} x_n}{\sum_n \tau_n^{k(t)}}$$

$$\Sigma_k^* = \arg\max \langle l(\boldsymbol{\theta}) \rangle, \qquad \Rightarrow \quad \Sigma_k^{(t+1)} = \frac{\sum_n \tau_n^{k(t)} (x_n - \mu_k^{(t+1)})(x_n - \mu_k^{(t+1)})^T}{\sum_n \tau_n^{k(t)}}$$

Fact:

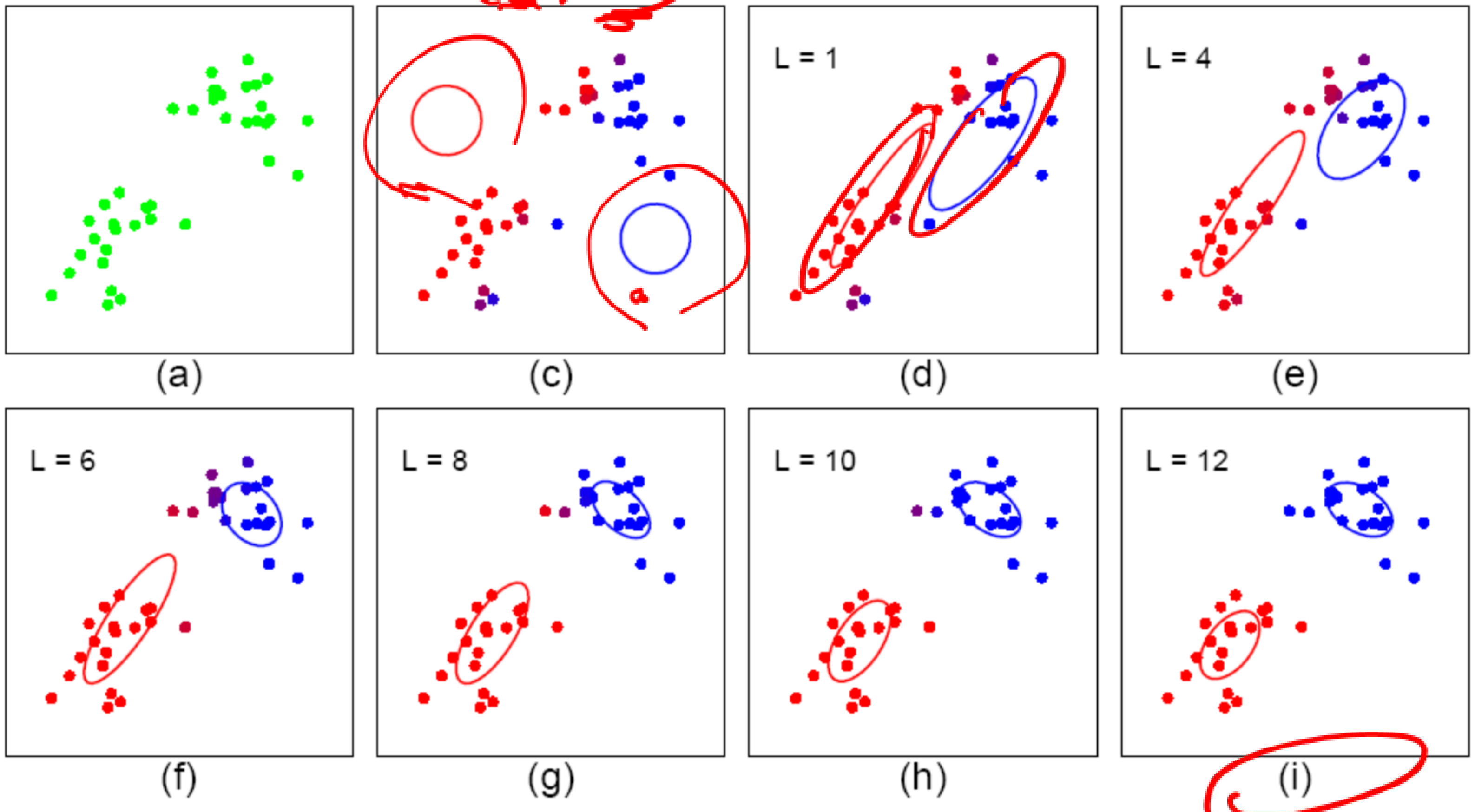
$$\frac{\partial \log |\mathrm{A}^{-1}|}{\partial \mathrm{A}^{-1}} = \mathrm{A}^T$$

$$\frac{\partial \mathbf{x}^T \mathrm{A} \mathbf{x}}{\partial \mathrm{A}} = \mathbf{x}\mathbf{x}^T$$

Example: Gaussian Mixture Models (GMMs)

- Start: "guess" the centroid μ_k and covariance Σ_k of each of the K clusters
- Loop:

$q(z|x)$



$\{<>\}$

$\theta_d = [0.5, 0.3, 0.2]$

topic model

pLSA

LDA

θ_d

Text Embedding

(distributed repr.) → $V_{king} - V_{man} + V_{woman} = V_{queen}$

Word Embedding

- Conventional word embedding:
  - Word2vec, Glove
  - A pre-trained **matrix**, each row is an embedding vector of a word

CDH 2003

→ 2023

2013

10-yr base paper

	0	1	2	3	4	5	6	7	8	9	...
fox	-0.348680	-0.077720	0.177750	-0.094953	-0.452890	0.237790	0.209440	0.037886	0.035064	0.899010	...
ham	-0.773320	-0.282540	0.580760	0.841480	0.258540	0.585210	-0.021890	-0.463680	0.139070	0.656720	...
brown	-0.374120	-0.076264	0.109260	0.186620	0.029943	0.182700	-0.631980	0.133060	-0.128980	0.603430	...
beautiful	0.171200	0.534390	-0.348540	-0.097234	0.101800	-0.170860	0.295650	-0.041816	-0.516550	2.117200	...
jumps	-0.334840	0.215990	-0.350440	-0.260020	0.411070	0.154010	-0.386110	0.206380	0.386700	1.460500	...
eggs	-0.417810	-0.035192	-0.126150	-0.215930	-0.669740	0.513250	-0.797090	-0.068611	0.634660	1.256300	...
beans	-0.423290	-0.264500	0.200870	0.082187	0.066944	1.027600	-0.989140	-0.259950	0.145960	0.766450	...
sky	0.312550	-0.303080	0.019587	-0.354940	0.100180	-0.141530	-0.514270	0.886110	-0.530540	1.556600	...
bacon	-0.430730	-0.016025	0.484620	0.101390	-0.299200	0.761820	-0.353130	-0.325290	0.156730	0.873210	...
breakfast	0.073378	0.227670	0.208420	-0.456790	-0.078219	0.601960	-0.024494	-0.467980	0.054627	2.283700	...
toast	0.130740	-0.193730	0.253270	0.090102	-0.272580	-0.030571	0.096945	-0.115060	0.484000	0.848380	...
today	-0.156570	0.594890	-0.031445	-0.077586	0.278630	-0.509210	-0.066350	-0.081890	-0.047986	2.803600	...
blue	0.129450	0.036518	0.032298	-0.060034	0.399840	-0.103020	-0.507880	0.076630	-0.422920	0.815730	...
green	-0.072368	0.233200	0.137260	-0.156630	0.248440	0.349870	-0.241700	-0.091426	-0.530150	1.341300	...
kings	0.259230	-0.854690	0.360010	-0.642000	0.568530	-0.321420	0.173250	0.133030	-0.089720	1.528600	...
dog	-0.057120	0.052685	0.003026	-0.048517	0.007043	0.041856	-0.024704	-0.039783	0.009614	0.308416	...
sausages	-0.174290	-0.064869	-0.046976	0.287420	-0.128150	0.647630	0.056315	-0.240440	-0.025094	0.502220	...
lazy	-0.353320	-0.299710	-0.176230	-0.321940	-0.385640	0.586110	0.411160	-0.418680	0.073093	1.486500	...
love	0.139490	0.534530	-0.252470	-0.125650	0.048748	0.152440	0.199060	-0.065970	0.128830	2.055900	...
quick	-0.445630	0.191510	-0.249210	0.465900	0.161950	0.212780	-0.046480	0.021170	0.417660	1.686900	...

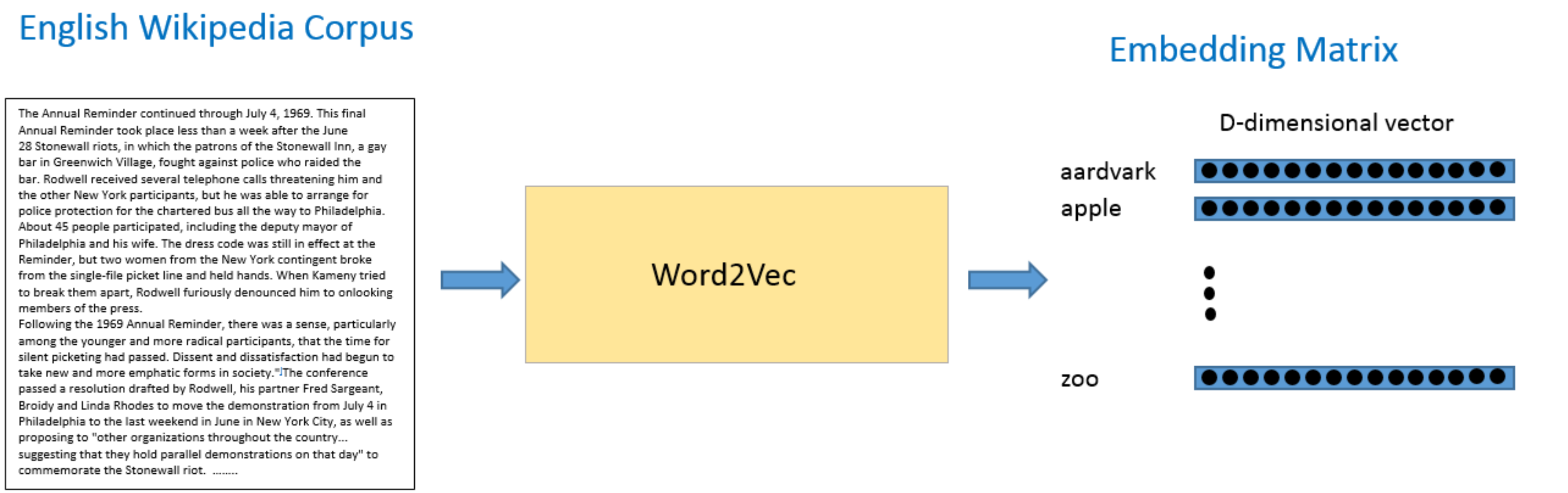
20 rows × 300 columns

[Courtesy: Vaswani, et al., 2017]

Word Embedding

- Conventional word embedding:
  - Word2vec, Glove
  - A pre-trained **matrix**, each row is an embedding vector of a word

	0	1	2	3	4	5	6	7	8	9	...
fox	-0.348680	-0.077720	0.177750	-0.094953	-0.452890	0.237790	0.209440	0.037886	0.035064	0.899010	...
ham	-0.773320	-0.282540	0.580760	0.841480	0.258540	0.585210	-0.021890	-0.463680	0.139070	0.658720	...
brown	-0.374120	-0.076264	0.109260	0.186620	0.029943	0.182700	-0.631980	0.133060	-0.128980	0.603430	...
beautiful	0.171200	0.534390	-0.348540	-0.097234	0.101800	-0.170860	0.295650	-0.041816	-0.516550	2.117200	...
jumps	-0.334840	0.215990	-0.350440	-0.260020	0.411070	0.154010	-0.386110	0.206380	0.386700	1.460500	...
eggs	-0.417810	-0.035192	-0.126150	-0.215930	-0.669740	0.513250	-0.797090	-0.068611	0.634660	1.256300	...
beans	-0.423290	-0.264500	0.200870	0.082187	0.066944	1.027600	-0.989140	-0.259950	0.145960	0.766450	...
sky	0.312550	-0.303080	0.019587	-0.354940	0.100180	-0.141530	-0.514270	0.886110	-0.530540	1.556600	...
bacon	-0.430730	-0.016025	0.484620	0.101390	-0.299200	0.761820	-0.353130	-0.325290	0.156730	0.873210	...
breakfast	0.073378	0.227670	0.208420	-0.456790	-0.078219	0.601960	-0.024494	-0.467980	0.054627	2.283700	...
							…945	-0.115060	0.484000	0.848380	...
							…350	-0.081890	-0.047986	2.803600	...
							…880	0.076630	-0.422920	0.815730	...
							…700	-0.091426	-0.530150	1.341300	...
							…250	0.133030	-0.089720	1.528600	...
							…704	-0.039783	0.009614	0.308416	...
							…315	-0.240440	-0.025094	0.502220	...
							…160	-0.418680	0.073093	1.486500	...
							…060	-0.065970	0.128830	2.055900	...
							…480	0.021170	0.417660	1.686900	...



[Image source: Va

Word Embedding

- **Problem:** word embeddings are applied in a context free manner

```
open a bank account          on the river bank

              [0.3, 0.2, -0.8, …]
```

Courtesy: Devlin 2019

Word Embedding

- **Problem:** word embeddings are applied in a context free manner

open a bank account   on the river bank

[0.3, 0.2, -0.8, …]

- **Solution:** Train contextual representations on text corpus

[0.9, -0.2, 1.6, …]   [-1.9, -0.4, 0.1, …]

open a bank account   on the river bank

Courtesy: Devlin 2019

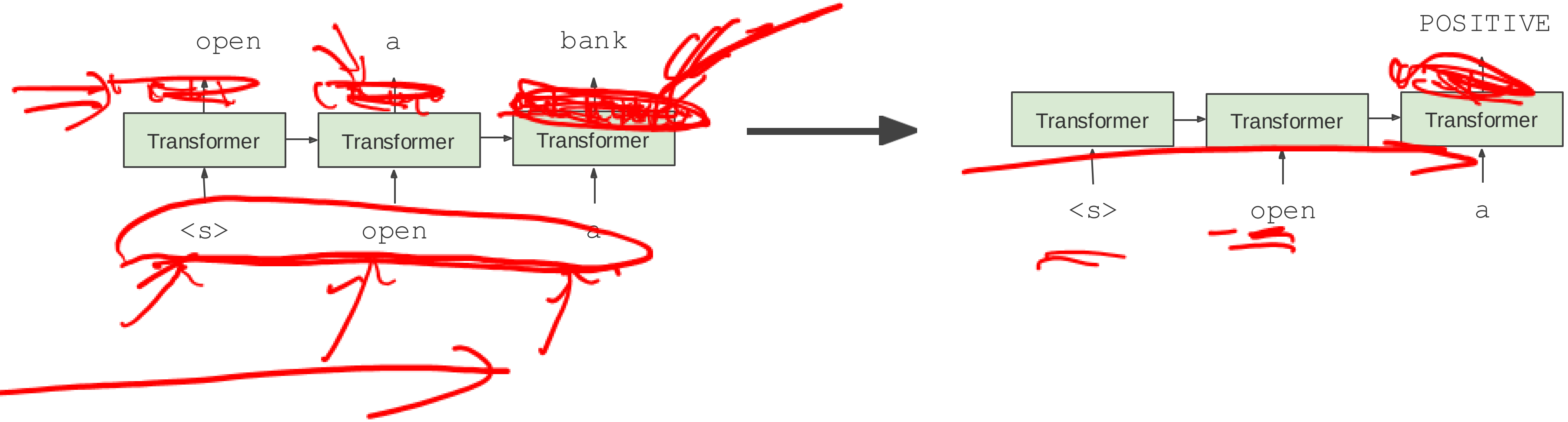
Contextual Representations

→ GPT-v1

GPT-4o

- *Improving Language Understanding by Generative Pre-Training*, OpenAI, 2018

Train Deep (12-layer) Transformer LM | **Fine-tune on Classification Task**



Courtesy: Devlin 2019

Problem with Previous Methods

- **Problem**: Language models only use left context *or* right context, but language understanding is bidirectional.

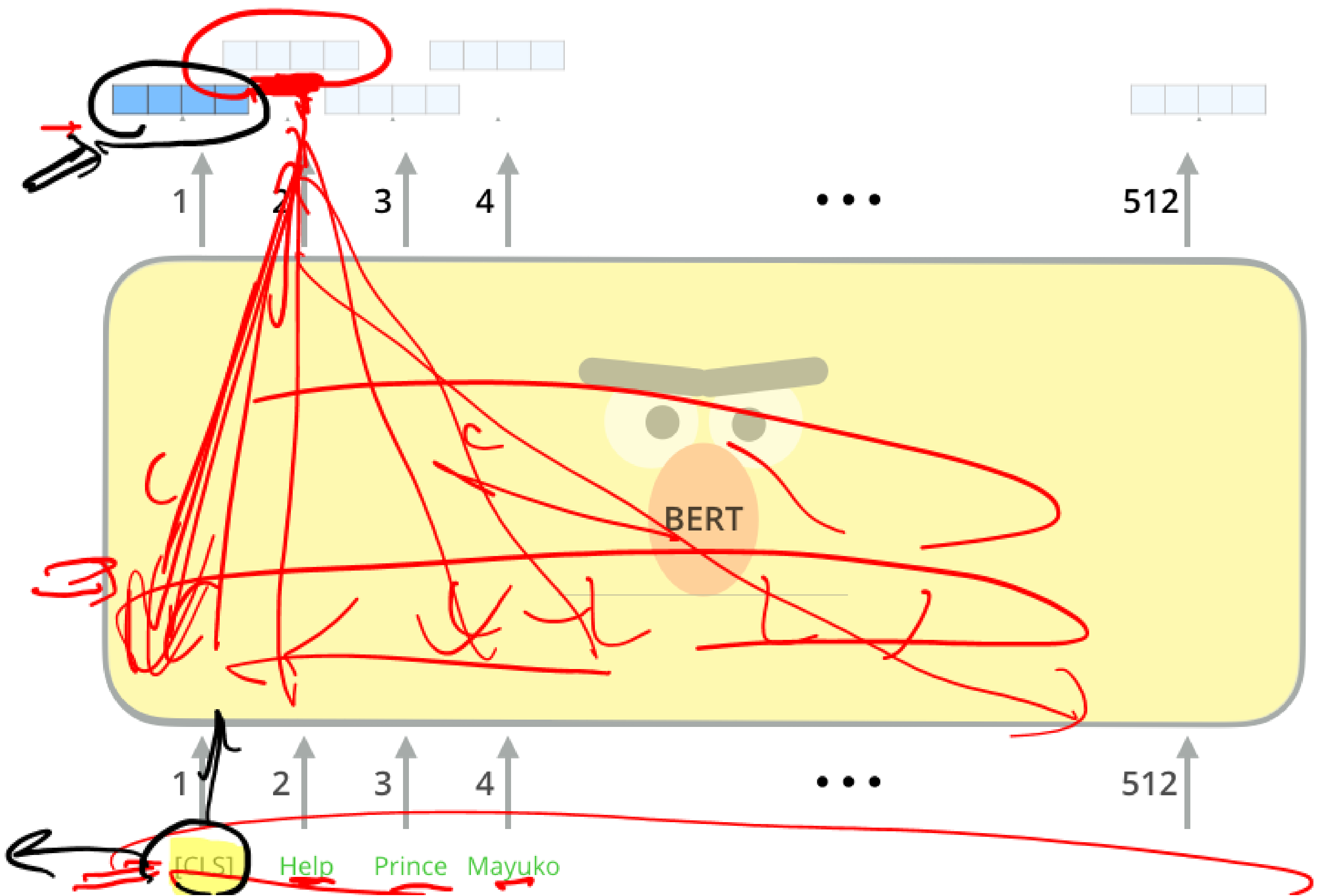
courtesy: Devlin 2019

2018

Pretraining

BERT

- BERT: A bidirectional model to extract contextual word embedding



BERT: Pre-training Procedure

- Dataset:
  - Wikipedia (2.5B words) + a collection of free ebooks (800M words)

BERT: Pre-training Procedure

- Dataset:
  - Wikipedia (2.5B words) + a collection of free ebooks (800M words)
- Training procedure
  - **masked language model** (masked LM)
    - Masks some percent of words from the input and has to reconstruct those words from context

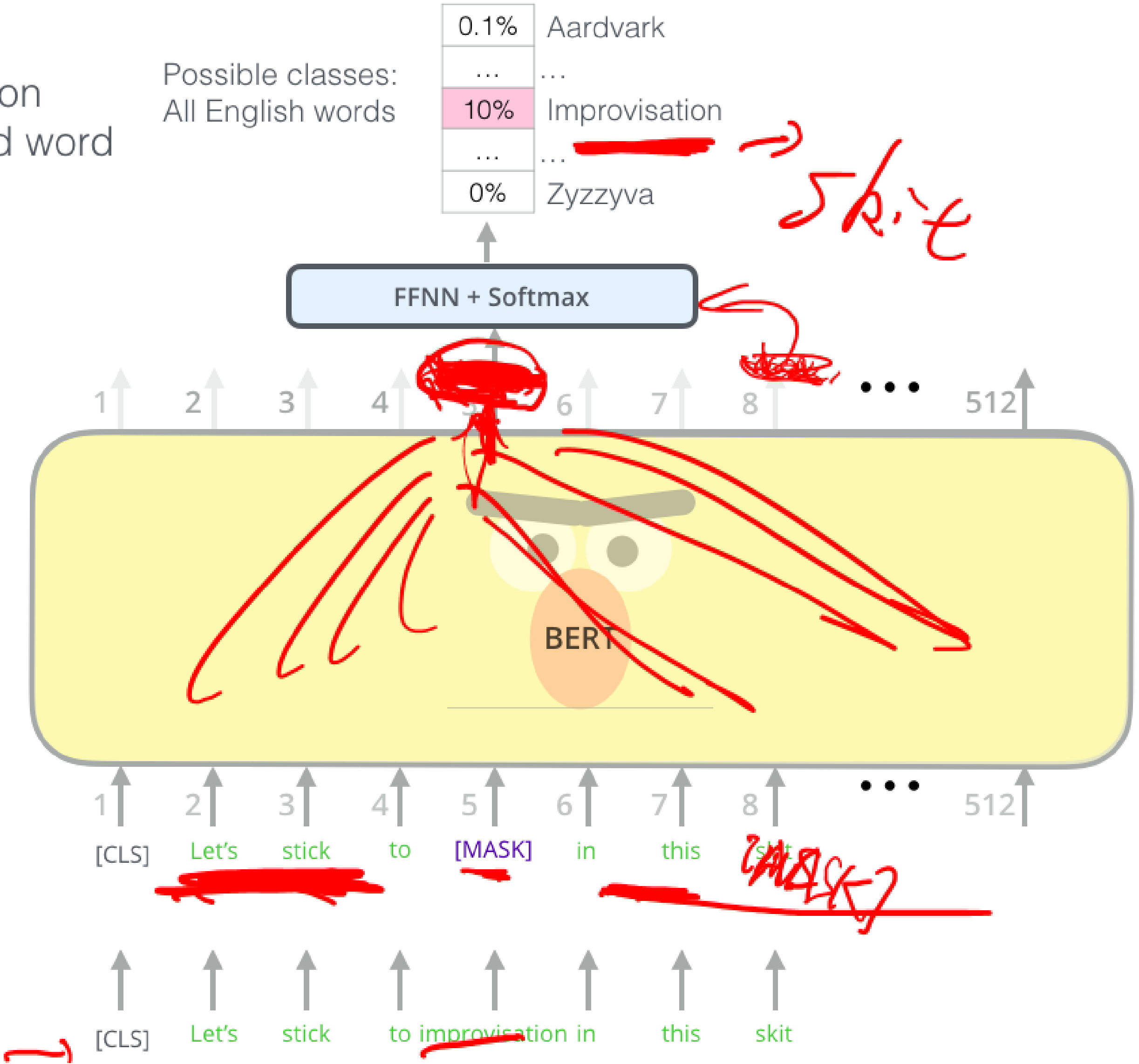
BERT: Pre-training Procedure

- Masked LM

Use the output of the masked word's position to predict the masked word

Randomly mask 15% of tokens

Input



BERT: Pre-training Procedure

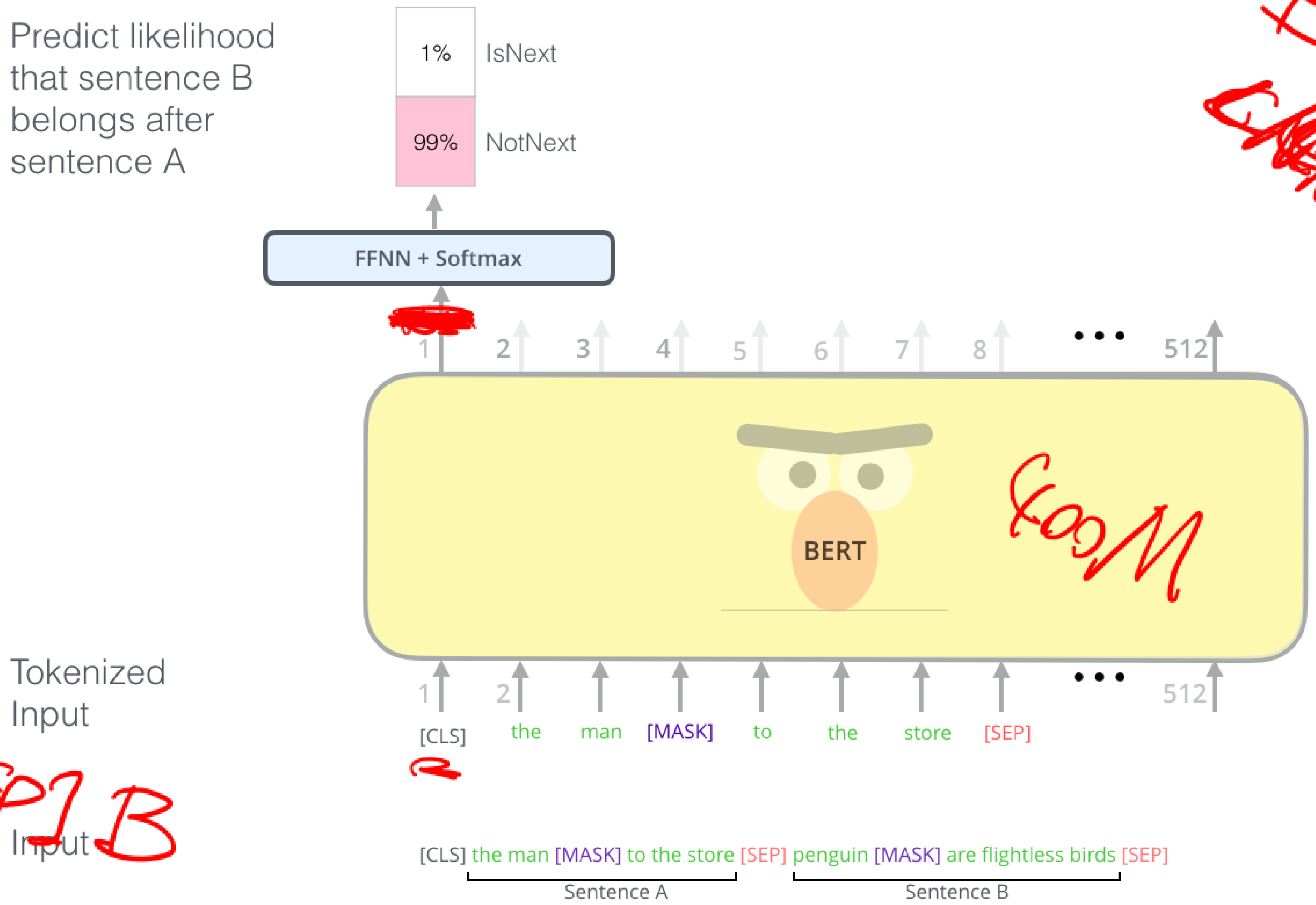
- Masked LM
- 15% masking:
    - Too little masking: Too expensive to train (few supervision signals per example)
    - Too much masking: Not enough context
- Problem: Mask token never seen at fine-tuning
- Solution: don't replace with [MASK] 100% of the time. Instead:
- 80% of the time, replace with [MASK]
    - went to the store → went to the [MASK]
- 10% of the time, replace random word
    - went to the store → went to the running
- 10% of the time, keep same
    - went to the store → went to the store

BERT: Pre-training Procedure

- Dataset:
  - Wikipedia (2.5B words) + a collection of free ebooks (800M words)
- Training procedure
  - **masked language model** (masked LM)
    - Masks some percent of words from the input and has to reconstruct those words from context
  - **Two-sentence task**
    - To understand relationships between sentences
    - Concatenate two sentences A and B and predict whether B actually comes after A in the original text

BERT: Pre-training Procedure

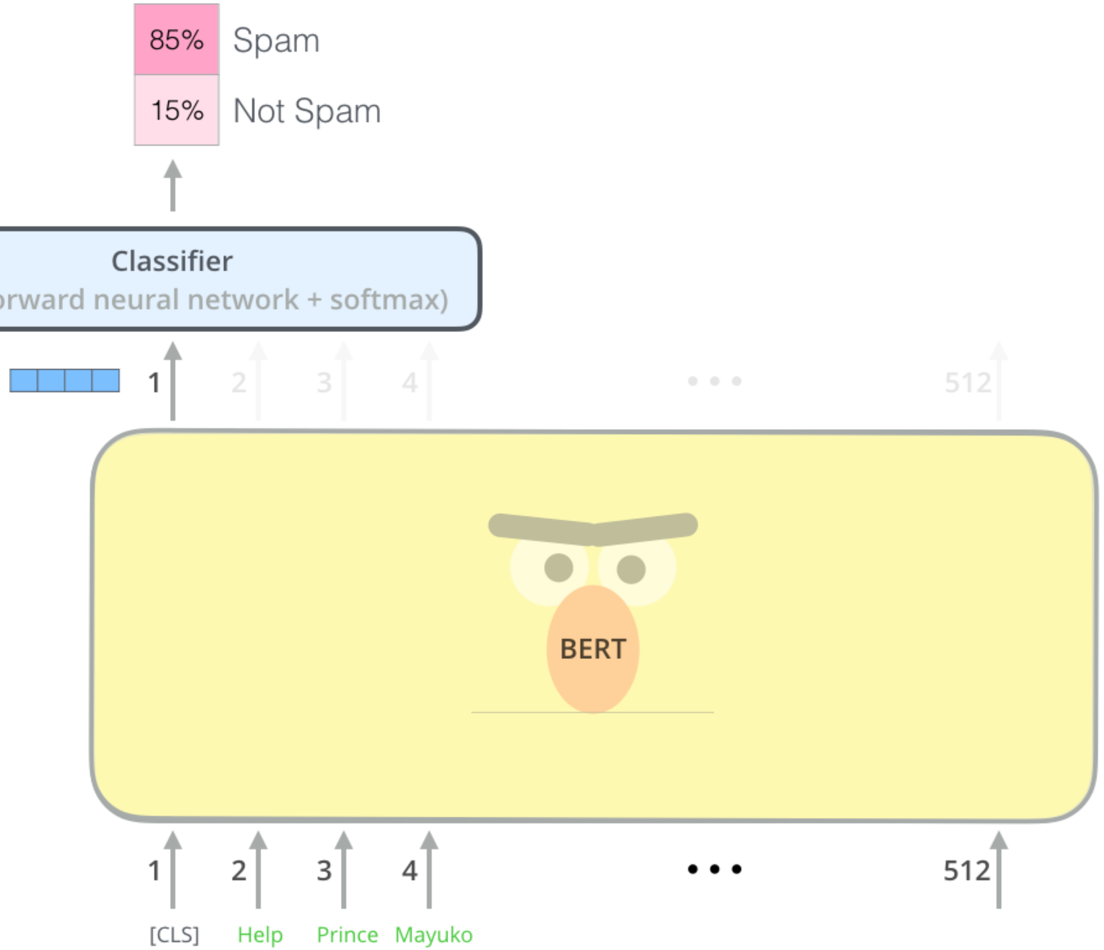
- Two sentence task



[CLS] A [SEP] B

BERT: Downstream Fine-tuning

- Use BERT for sentence classification

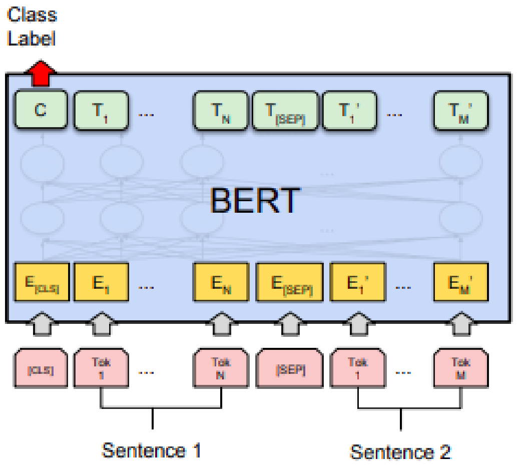


BERT: Downstream Fine-tuning

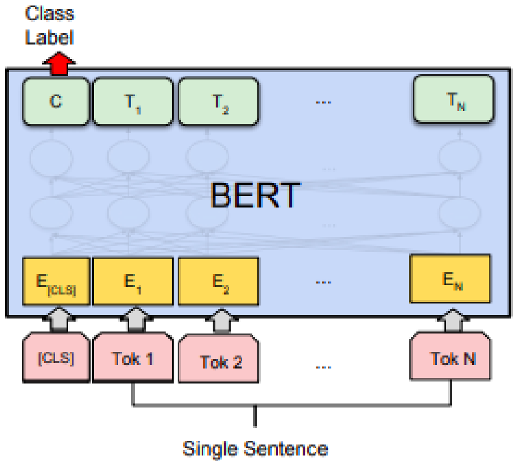
NER

NLI

STS

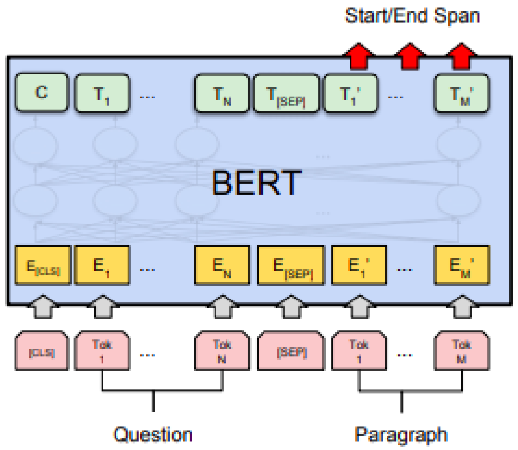


(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG

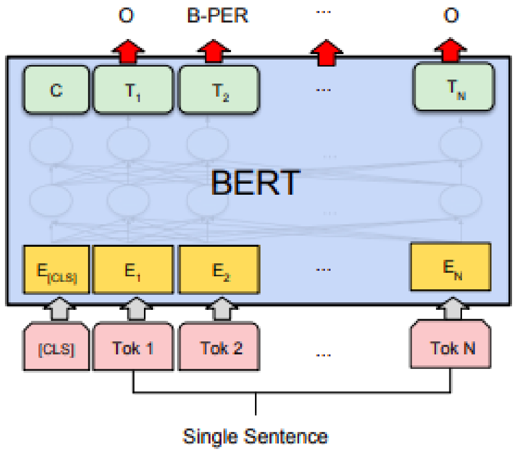


(b) Single Sentence Classification Tasks:
SST-2, CoLA

PER



(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

A B

Jay lives in SD

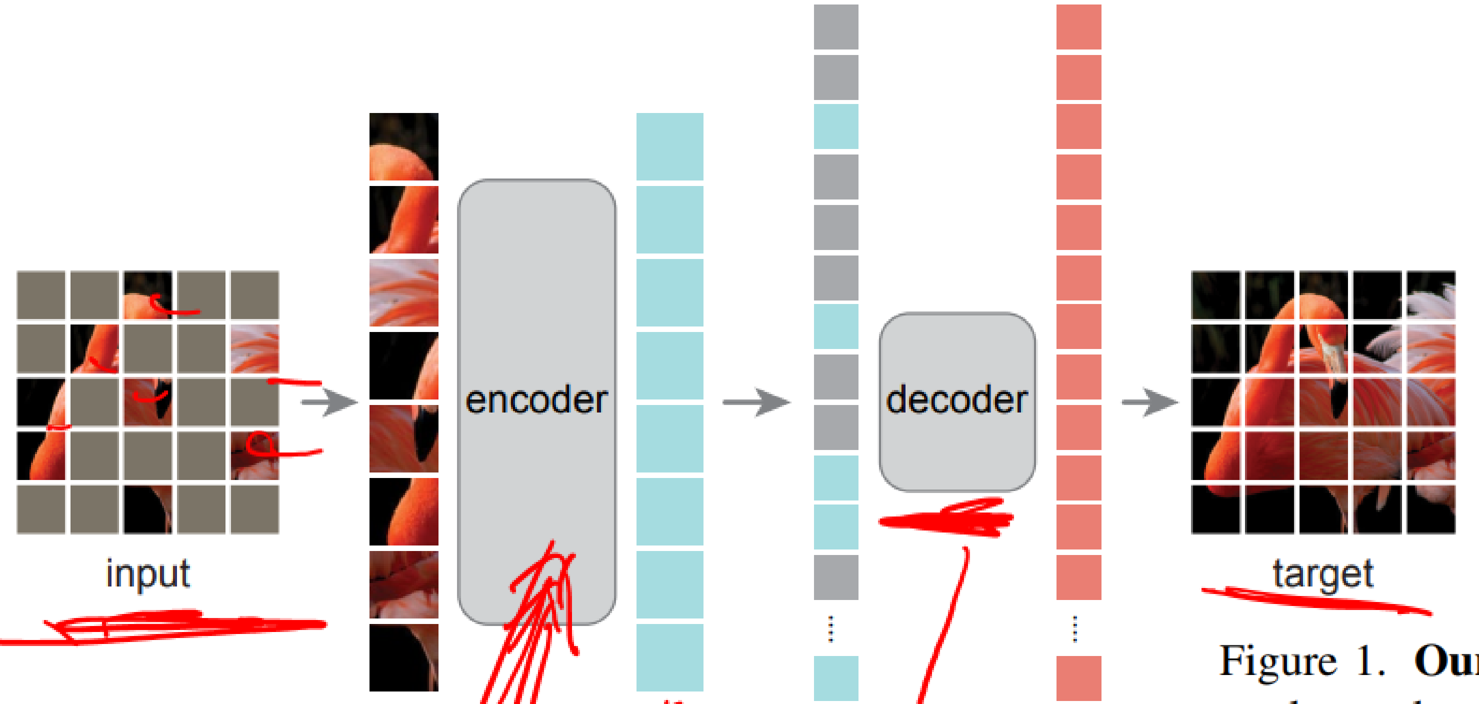
BERT Results

- Huge improvements over SOTA on 12 NLP task

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

Table 1: GLUE Test results, scored by the GLUE evaluation server. The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set. OpenAI GPT = (L=12, H=768, A=12); $\text{BERT}_{\text{BASE}}$ = (L=12, H=768, A=12); $\text{BERT}_{\text{LARGE}}$ = (L=24, H=1024, A=16). BERT and OpenAI GPT are single-model, single task. All results obtained from https://gluebenchmark.com/leaderboard and https://blog.openai.com/language-unsupervised/.

Similar idea for image embedding: masked autoencoder (MAE)



BERT

~~classification~~

Figure 1. **Our MAE architecture**. During pre-training, a large random subset of image patches (*e.g.*, 75%) is masked out. The encoder is applied to the small subset of *visible patches*. Mask tokens are introduced *after* the encoder, and the full set of encoded patches and mask tokens is processed by a small decoder that reconstructs the original image in pixels. After pre-training, the decoder is discarded and the encoder is applied to uncorrupted images (full sets of patches) for recognition tasks.

[He et al., 2021: Masked Autoencoders Are Scalable Vision Learners]

Similar idea for image embedding: masked autoencoder (MAE)

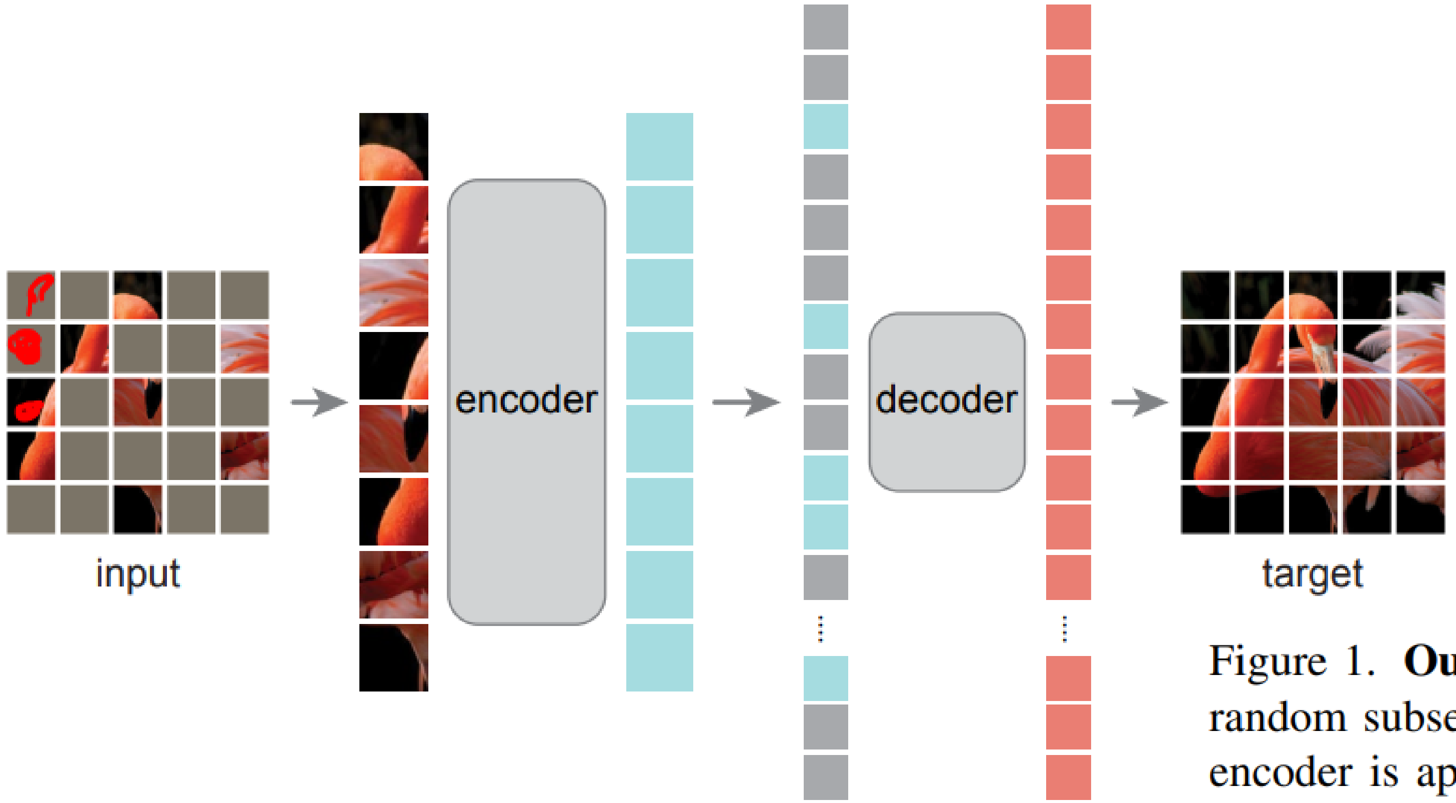


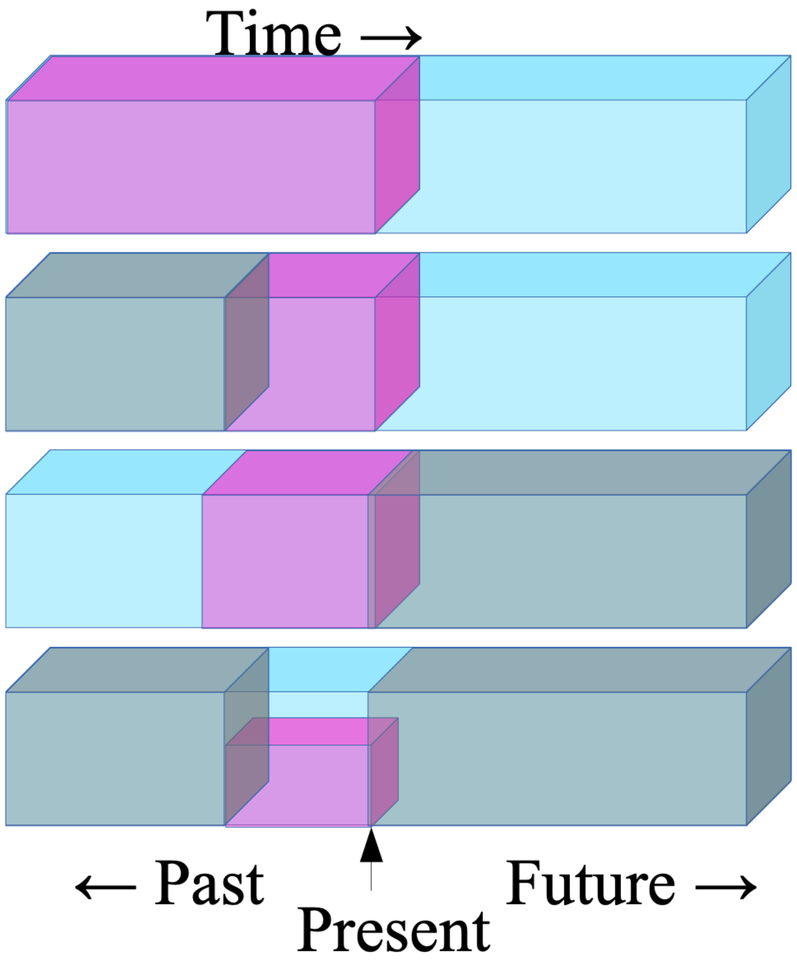
Figure 1. **Our MAE architecture**. During pre-training, a large random subset of image patches (*e.g.*, 75%) is masked out. The encoder is applied to the small subset of *visible patches*. Mask tokens are introduced *after* the encoder, and the full set of encoded patches and mask tokens is processed by a small decoder that reconstructs the original image in pixels. After pre-training, the decoder is discarded and the encoder is applied to uncorrupted images (full sets of patches) for recognition tasks.

Question: Why is this (75%) much larger than the mask rate in BERT (15%)?

[He et al., 2021: Masked Autoencoders Are Scalable Vision Learners]

More general idea: Self-Supervised Learning

- Predict any part of the input from any other part.
- Predict the future from the past.
- Predict the future from the recent past.
- Predict the past from the present.
- Predict the top from the bottom.



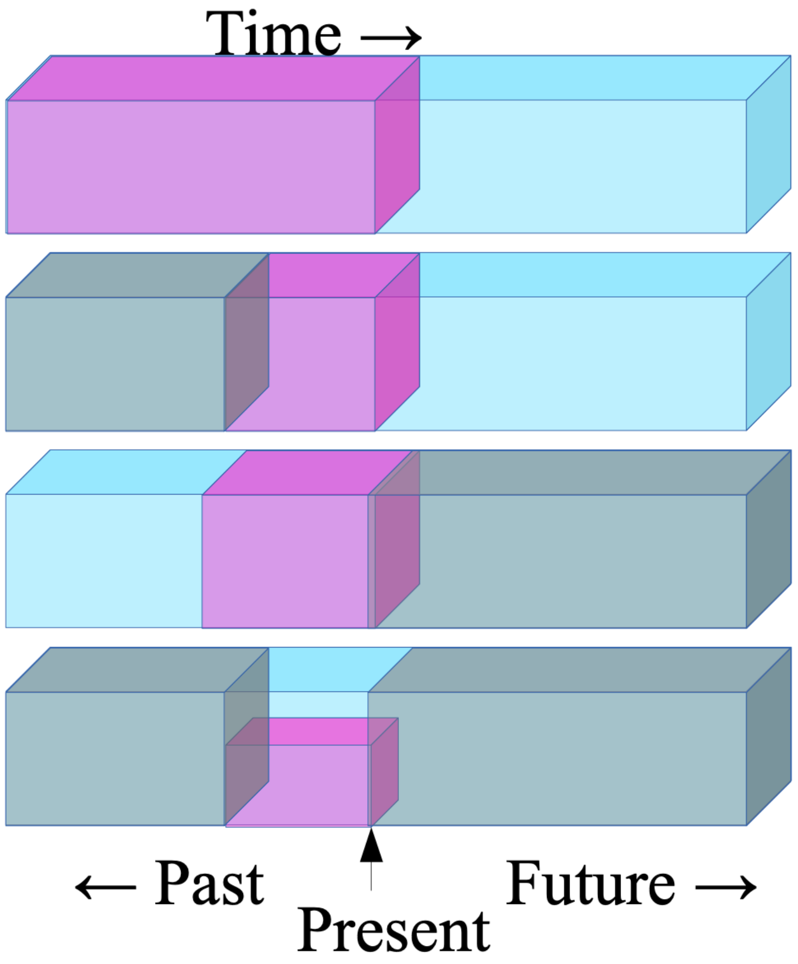
un-supervised-self

2018

[Courtesy: Lecun "Self-supervised Learning"]

More general idea: Self-Supervised Learning

- **Predict any part of the input from any other part.**
- **Predict the future from the past.**
- **Predict the future from the recent past.**
- **Predict the past from the present.**
- **Predict the top from the bottom.**
- **Predict the ~~occluded from the visible~~**
- **Pretend there is a part of the input you don't know and predict that.**



[Courtesy: Lecun "Self-supervised Learning"]

More general idea: Self-Supervised Learning: Motivation (I)

- Our brains do this all the time

- Filling in the visual field at the retinal blind spot
- Filling in occluded images, missing segments in speech
- Predicting the state of the world from partial (textual) descriptions
- Predicting the consequences of our actions
- Predicting the sequence of actions leading to a result

- Predicting any part of the past, present or future percepts from whatever information is available.

[Courtesy: Lecun "Self-supervised Learning"]

More general idea: Self-Supervised Learning: Motivation (I)

- Successfully learning to predict everything from everything else would result in the accumulation of lots of background knowledge about how the world works

- The model is forced to learn what we really care about, e.g. a semantic representation, in order to solve the prediction problem

[Courtesy: Lecun “Self-supervised Learning”]

[Courtesy: Zisserman “Self-supervised Learning”]

More general idea: Self-Supervised Learning: Motivation (II)

- The machine predicts any part of its input from any observed part
    - **A lot of** supervision signals in each data instance

- Untapped/availability of vast numbers of unlabeled text/images/videos..
    - Facebook: one billion images uploaded per day
    - 300 hours of video are uploaded to YouTube every minute

cos trillions of text [illegible]

[Courtesy: Zisserman “Self-supervised Learning”]

Questions?