

STOR566: Introduction to Deep Learning

Lecture 11: NLP Pre-training

Yao Li
UNC Chapel Hill

Sep 26, 2024

Materials are from *Deep Learning (UCLA)*

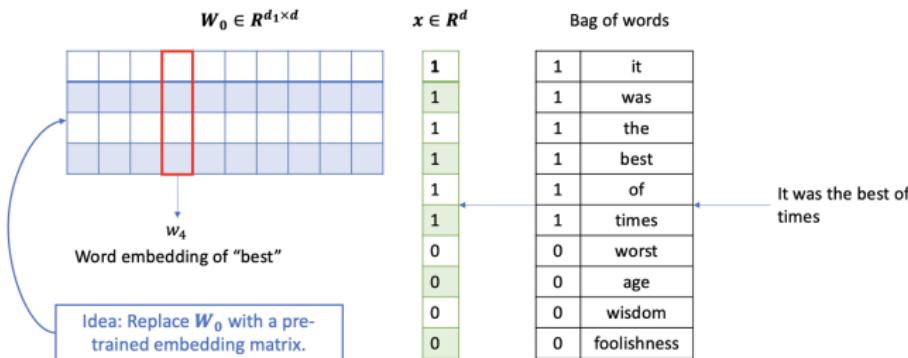
Unsupervised pretraining for NLP

Motivation

- Many unlabeled NLP data but very few labeled data
- Can we use large amount of unlabeled data to obtain meaningful representations of words/sentences?

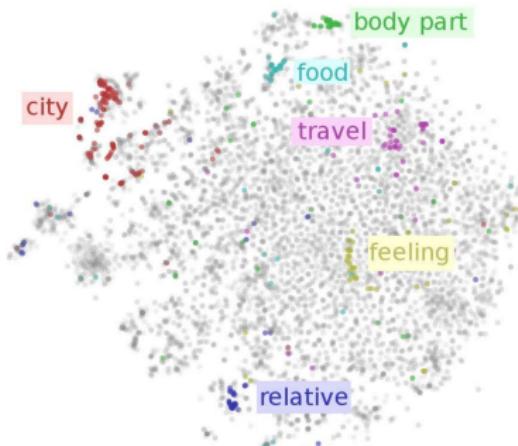
Learning word embeddings

- Use large (unlabeled) corpus to learn a useful word representation
 - Learn a vector for each word based on the corpus
 - Hopefully the vector represents some semantic meaning
 - Can be used for many tasks
 - Replace the word embedding matrix for DNN models for classification/translation



Methods for NLP Pre-training

- Two different perspectives but led to similar results:
 - Word2vec (Mikolov et al., 2013)
 - Glove (Pennington et al., 2014)
- Other Methods:
 - PPMI (Levy et al., 2014)
 - CoVe (McCann et al., 2017)
 - ELMo (Peter et al., 2018)
- New Trend:
 - BERT (Devlin et al., 2019)
 - CLIP (Radford et al., 2021)



Context information

- Given a large text corpus, how to learn low-dimensional features to represent a word?
- For each word w_i , define the “contexts” of the word as the words surrounding it in an L -sized window:

$$w_{i-L-2}, w_{i-L-1}, \underbrace{w_{i-L}, \dots, w_{i-1}}_{\text{contexts of } w_i}, w_i, \underbrace{w_{i+1}, \dots, w_{i+L}}_{\text{contexts of } w_i}, w_{i+L+1}, \dots$$

- Get a collection of (word, context) pairs, denoted by D .

Word pair

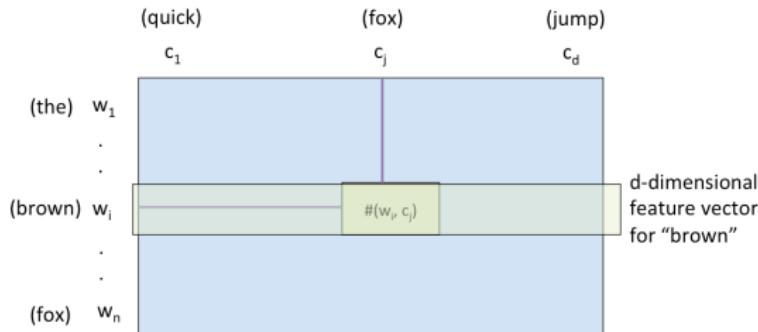
Source Text	Training Samples
The quick brown fox jumps over the lazy dog. ➔	(the, quick) (the, brown)
The quick brown fox jumps over the lazy dog. ➔	(quick, the) (quick, brown) (quick, fox)
The quick brown fox jumps over the lazy dog. ➔	(brown, the) (brown, quick) (brown, fox) (brown, jumps)
The quick brown fox jumps over the lazy dog. ➔	(fox, quick) (fox, brown) (fox, jumps) (fox, over)

Figure from <http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/>

Use bag-of-word model

- Idea 1: Use the bag-of-word model to “describe” each word
- Assume we have context words c_1, \dots, c_d in the corpus, compute
$$\#(w, c_i) := \text{number of times the pair } (w, c_i) \text{ appears in } D$$
- For each word w , form a d -dimensional (sparse) vector to describe w

$$\#(w, c_1), \dots, \#(w, c_d),$$



PMI/PPMI Representation

- Similar to TF-IDF: Need to consider the frequency of each word and each context
- Instead of using co-ocurrent count $\#(w, c)$, we can define pointwise mutual information:

$$\text{PMI}(w, c) = \log\left(\frac{\hat{P}(w, c)}{\hat{P}(w)\hat{P}(c)}\right) = \log \frac{\#(w, c)|D|}{\#(w)\#(c)},$$

- $\#(w) = \sum_c \#(w, c)$: number of pairs with word w
- $\#(c) = \sum_w \#(w, c)$: number of pairs with word c
- $|D|$: number of pairs in D

PMI/PPMI Representation

- Similar to TF-IDF: Need to consider the frequency of each word and each context
- Instead of using co-ocurrent count $\#(w, c)$, we can define pointwise mutual information:

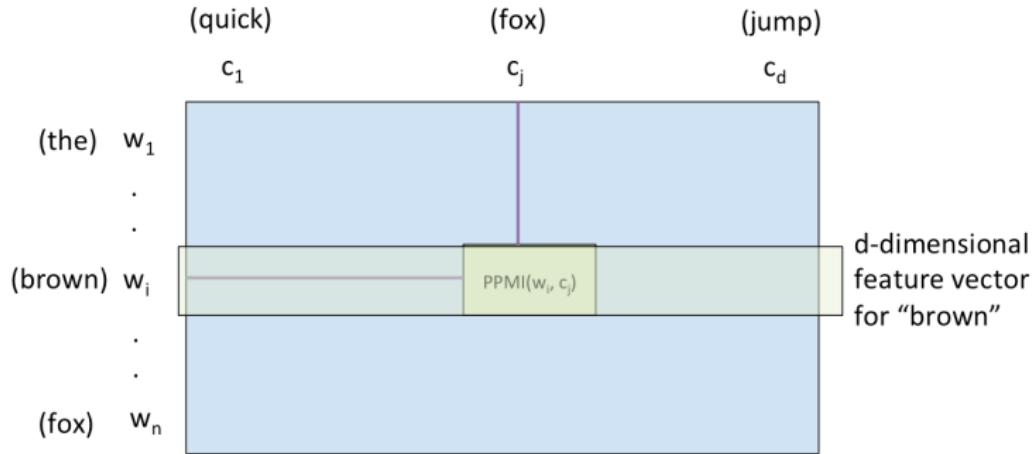
$$\text{PMI}(w, c) = \log\left(\frac{\hat{P}(w, c)}{\hat{P}(w)\hat{P}(c)}\right) = \log \frac{\#(w, c)|D|}{\#(w)\#(c)},$$

- $\#(w) = \sum_c \#(w, c)$: number of pairs with word w
- $\#(c) = \sum_w \#(w, c)$: number of pairs with word c
- $|D|$: number of pairs in D
- Positive PMI (PPMI) usually achieves better performance:

$$\text{PPMI}(w, c) = \max(\text{PMI}(w, c), 0)$$

- M^{PPMI} : word feature matrix with $\text{PPMI}(w, c)$ as element

PPMI Matrix



Low-dimensional embedding

- Perform PCA/SVD on the sparse feature matrix:

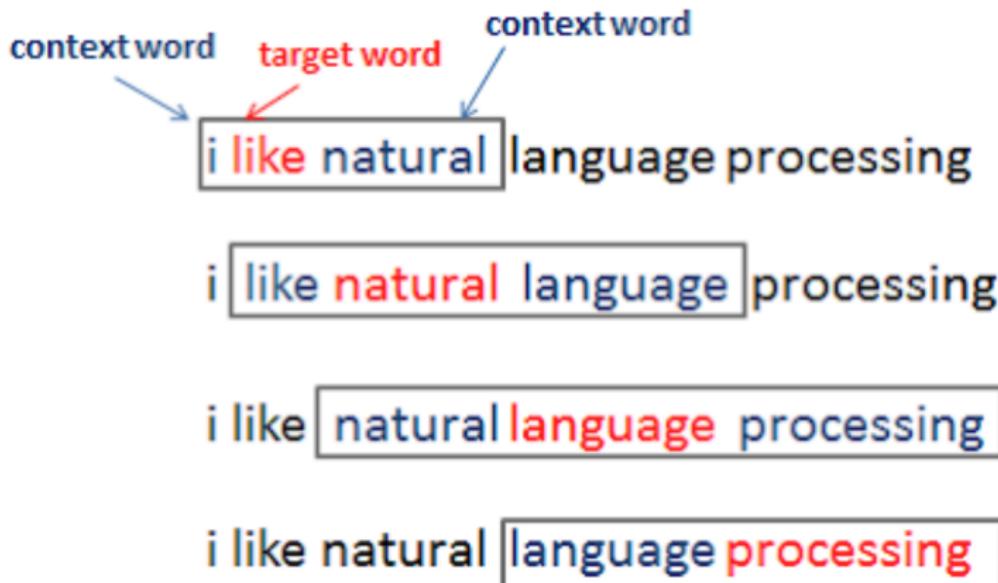
$$M^{\text{PPMI}} \approx U_k \Sigma_k V_k^T$$

Then $W^{\text{SVD}} = U_k \Sigma_k$ is the context representation of each word
(Each row is a k -dimensional feature for a word)

- $k \ll d$

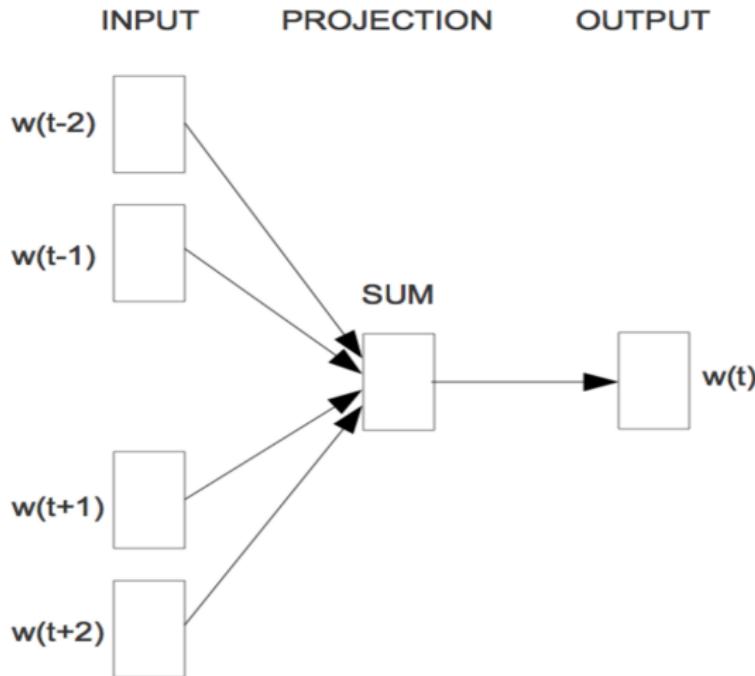
Word2vec (Mikolov et al., 2013)

- A neural network model for learning word embeddings
- Main idea:
 - Predict the target words based on the neighbors (CBOW)
 - Predict neighbors given the target words (Skip-gram)



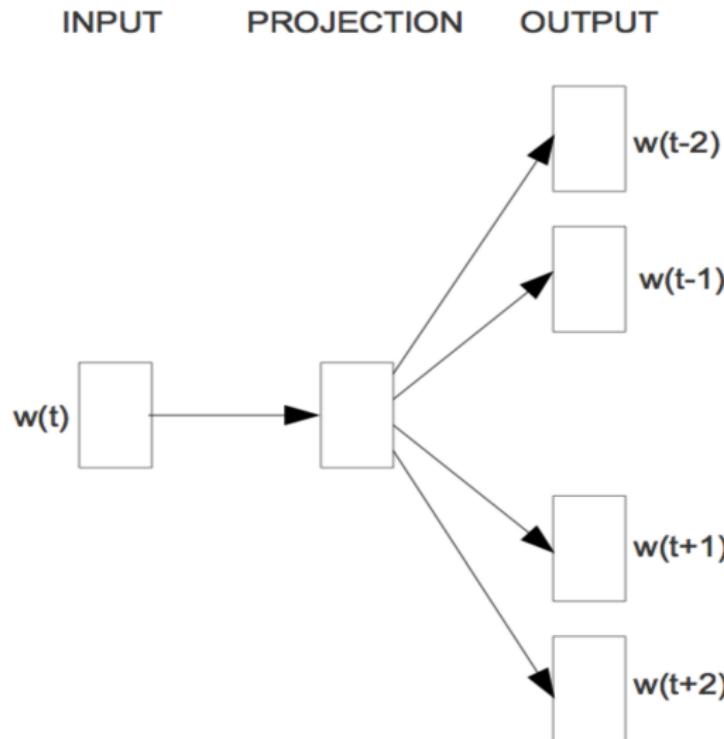
CBOW

- Predict the target words based on the neighbors



Skip-gram

- Predict neighbors using target word



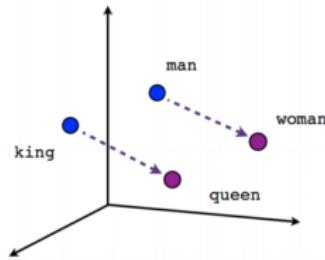
More on skip-gram

- Learn the probability $P(w_{t+j}|w_t)$: the probability to see w_{t+j} in target word w_t 's neighborhood
- Every word has two embeddings:
 - v_i serves as the role of target
 - u_i serves as the role of context
- Model probability as softmax:

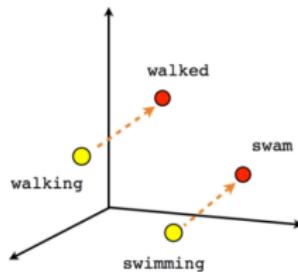
$$P(o|c) = \frac{e^{u_o^T v_c}}{\sum_{w=1}^W e^{u_w^T v_c}}$$

Results

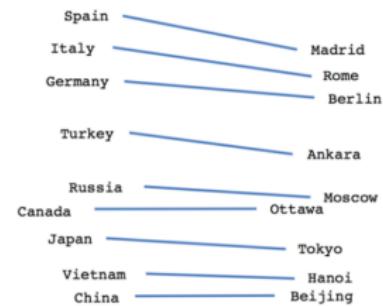
The low-dimensional embeddings are (often) meaningful:



Male-Female



Verb tense



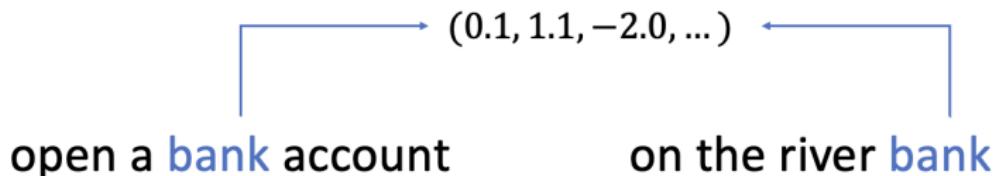
Country-Capital

Figure from <https://www.tensorflow.org/tutorials/word2vec>

Contextual embedding

Contextual word representation

- Non-contextual word embedding:

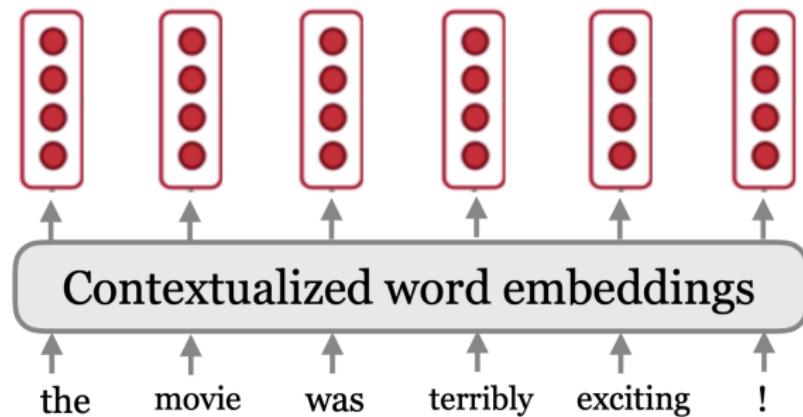


- The semantic meaning of a word should depend on its context



Contextual word representation

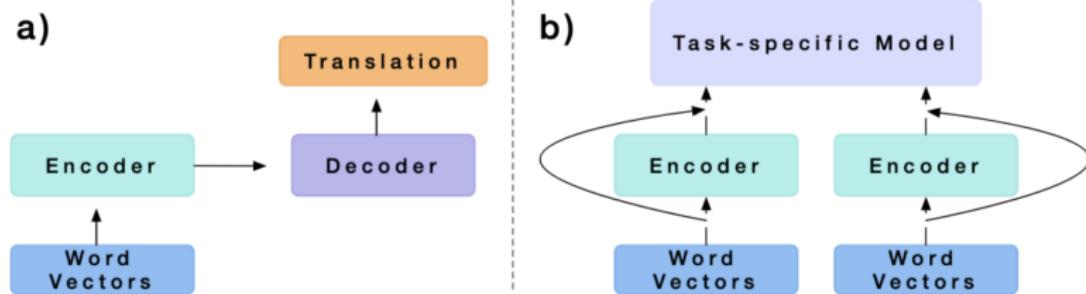
- Solution: Train a model to extract contextual representations on text corpus



- Share the model instead of the fixed embedding

CoVe (McCann et al., 2017)

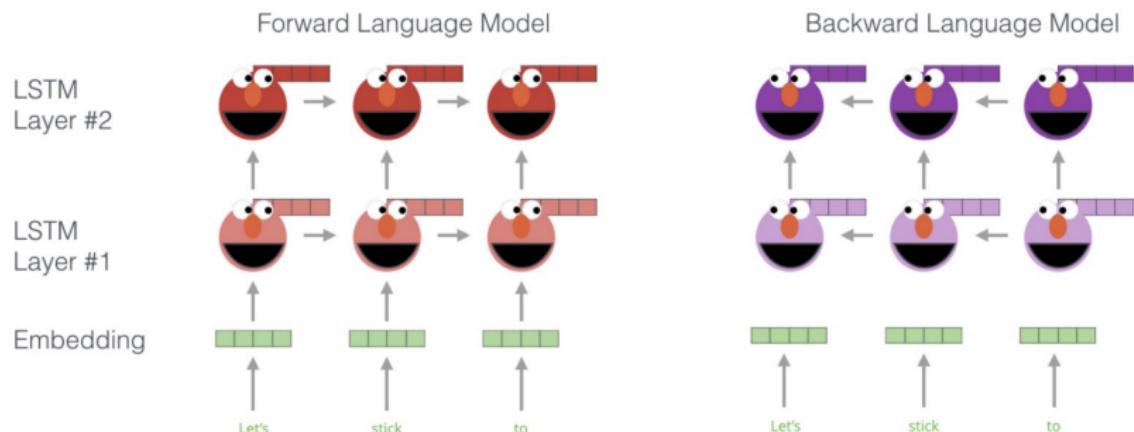
- Key idea: Train a standard neural machine translation model
- Take the encoder directly as contextualized word embeddings



- Bi-directional LSTM

ELMo (Peter et al., 2018)

- Key ideas:
 - Train a bidirectional LSTM language model on large corpus
 - Use the hidden states for each token to compute a vector representation of each word
 - Replace the word embedding by Elmo's embedding (with fixed Elmo's LSTM weights)



ELMo results

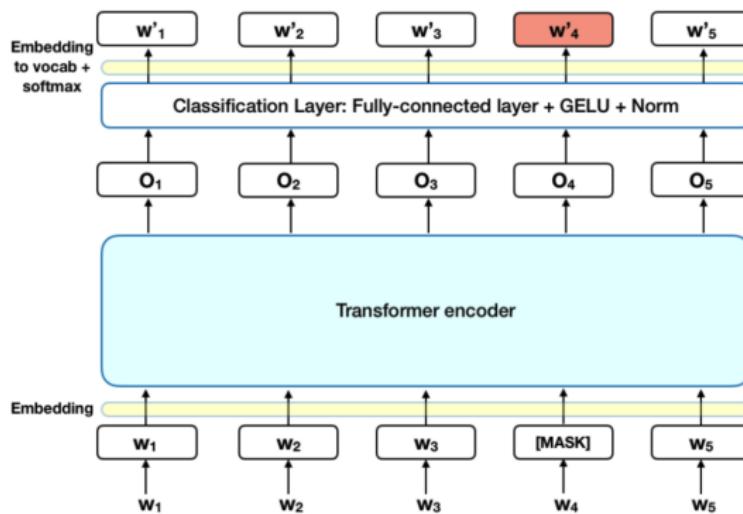
TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (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%

BERT

- Key ideas: replace LSTM by Transformer
- Define the generated pretraining task by masked language model
- Two pretraining tasks
- Finetune both BERT weights and task-dependent model weights for each task

BERT pretraining loss

- Masked language model: predicting each word by the rest of sentence
- Next sentence prediction: the model receives pairs of sentences as input and learns to predict if the second sentence is the subsequent sentence in the original document.

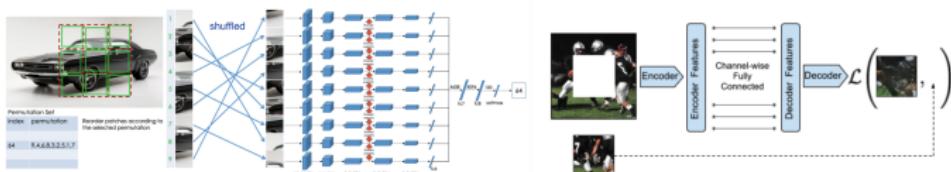
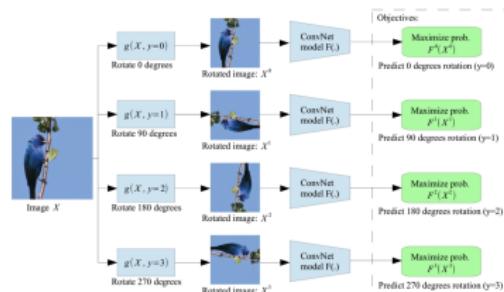


BERT results

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
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	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	82.1

Unsupervised pretraining for Vision

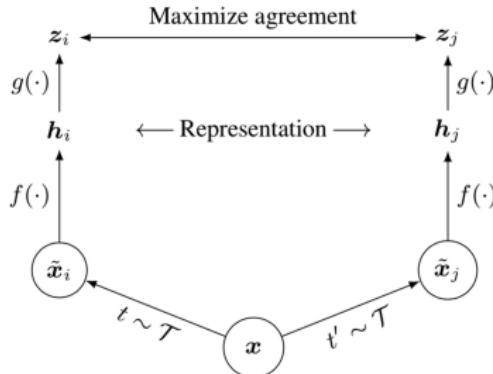
- Pretrain image encoders without using labels
- Many “pretraining tasks” have been proposed
 - Rotation, coloring, jigsaw puzzle, generative modeling



Contrastive learning

- Learns representations of visual inputs by maximizing agreement between differently augmented views of the same sample (SimCLR, Chen et al., 2020)
 - Randomly sample a batch of images, for each x generate \tilde{x}_i, \tilde{x}_j based on different augmentations
 - Pass each image into encoder $z_i = g(f(\tilde{x}_i))$
 - Contrastive loss:

$$L_{i,j} = -\log \frac{e^{\text{sim}(z_i, z_j)/\tau}}{\sum_{k \neq i} e^{\text{sim}(z_i, z_k)/\tau}}$$



Conclusions

- PPMI
- Word2vec
- Contextual embedding

Questions?