CS 5740: Natural Language Processing

Contextualized Representations

Instructor: Yoav Artzi

Slides adapted from Greg Durrett

Overview

- Motivation
- Context-dependent Representations with BERT
- Tokenization for BERT (and elsewhere)
- Common usage recipes
- Examples of less common uses:
    - Cross-modality representations
    - Generation evaluation with BERT

Motivation

- Word embeddings (e.g., word2vec, GloVe):
  - Learn a vector for each word type
  - Always the same vector
- Problem: each vector likely mixes multiple senses, regardless of specific instance use

Motivation

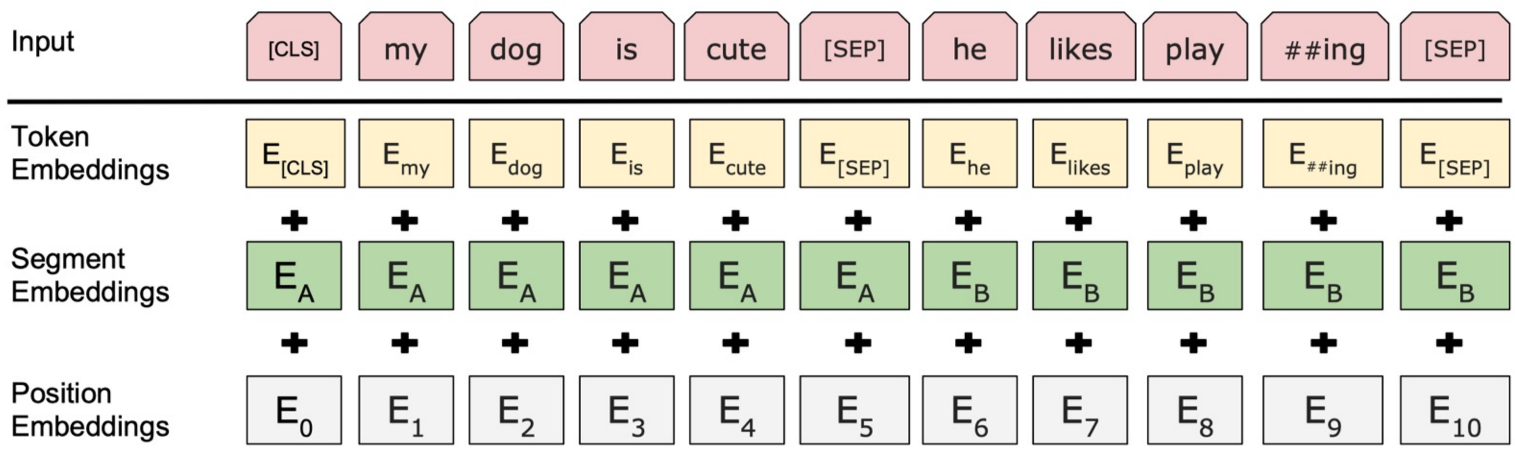
- Instead of a single vector: learn a different vector for each use of a word type
- Challenge: how do we define the space of uses? Isn’t it too large?
- Solution: use sentence encoders to create a custom vector for every instance of a word

Several Approaches

- Central Word Prediction Objective (context2vec) [Melamud et al. 2016]
- Machine Translation Objective (CoVe) [McMann et al. 2017]
- Bi-directional Language Modeling Objective (ELMo) [Peters et al. 2018]
- Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al. 2018]
- Robustly Optimized BERT (RoBERTa) [Liu et al. 2019]
- And more and more …

BERT

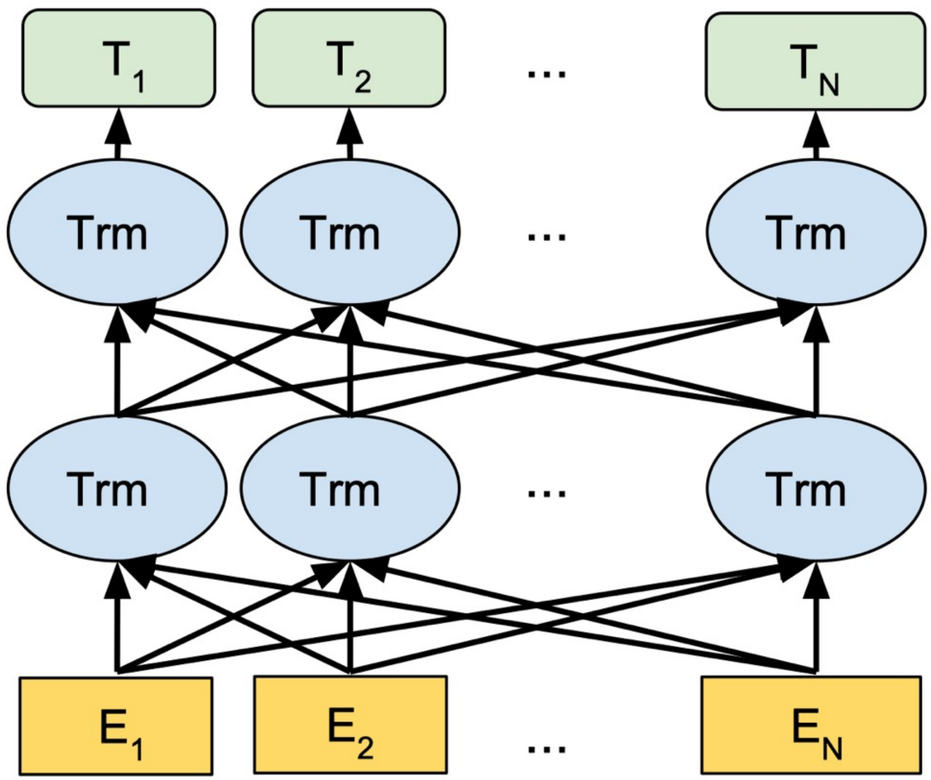
- Input: a sentence or a pair of sentences with a separator and subword representation
- Why do we need positional embedding?



[figure from Devlin et al. 2018]

BERT

- Model: multi-layer self-attention (Transformer)



[figure from Devlin et al. 2018]

BERT

- Model: multi-layer self-attention (Transformer)
- BERT Base: 12 layers, 768-dim per word-piece token, 12 heads. Total parameters = 110M
- BERT Large: 24 layers, 1024-dim per word-piece token, 16 heads. Total parameters = **340M**
- RoBERTa: same model, much more data (160GB of data instead of 16GB)

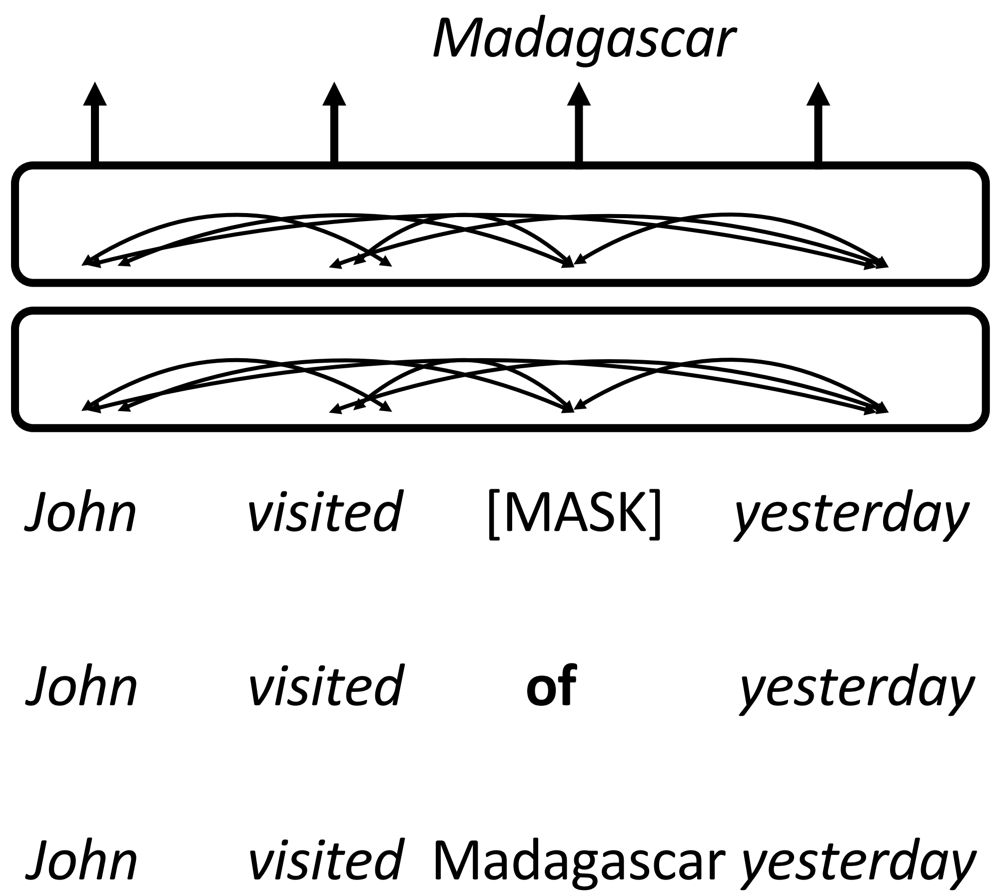
Training BERT

- Key idea: self-supervised objectives with raw text
- Two objectives: masked language modeling and next sentence prediction
- Data: BookCorpus + English Wikipedia
- Later development with RoBERTa:
  - Much more data
  - Removed the next sentence prediction objective
  - Dynamic masking

Masked Language Modeling

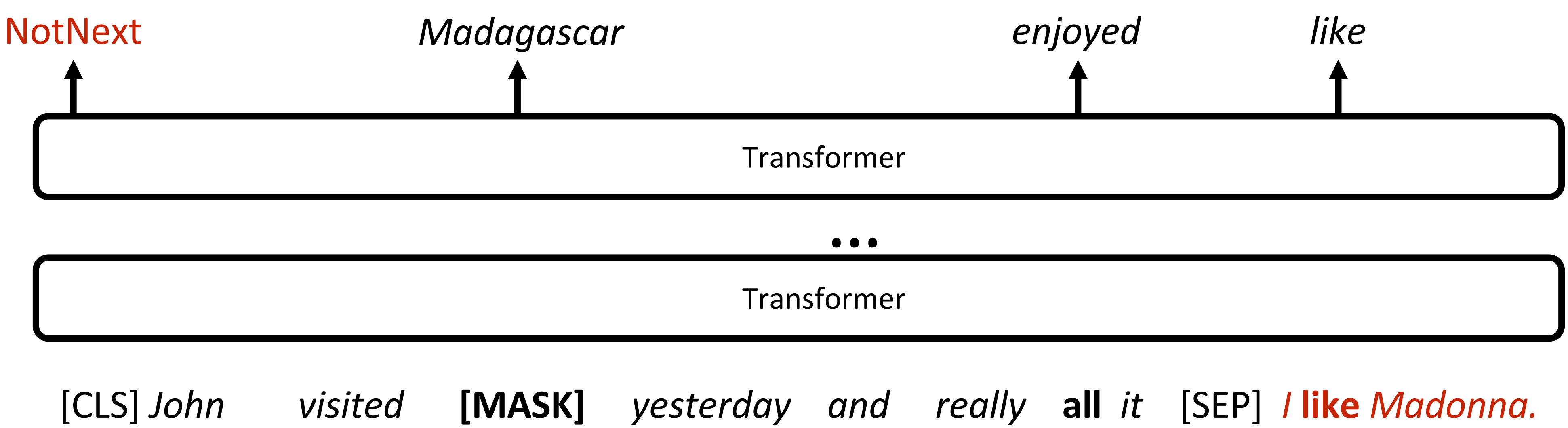
- Similar to predicting the next word for language modeling, but adapted for non-directional self-attention

- The BERT recipe: mask and predict 15% of the tokens
  - For 80% (of 15%) replace with the input token with [MASK]
  - For 10%, replace with a random token
  - For 10%, keep the same



Next Sentence Prediction

- Input: [CLS] Text chunk 1 [SEP] Text chunk 2
- Create data: 50% of the time, take the true next chunk of text, 50% of the time take a random other chunk
- Predict whether the next chunk is the “true” next



Sub-word Tokenization

- BERT uses Word Piece tokenization
- Related models (e.g., for MT, language modeling, etc) use either Word Piece or Byte Pair Encoding tokenization
- Advantage: no unknown words problem
- Package: https://github.com/huggingface/tokenizers

Byte Pair Encoding (BPE) Tokenization

1. Start with every individual byte (basically character) as its own token

2. Count bigram token cooccurrences over tokens (potentially: weight according to corpus frequencies)

3. Merge the most frequent pair of adjacent tokens to create a new token

- Vocabulary size is controlled by the number of merges

- With ~8000 tokens we get many whole words in English

[Sennrich et al. (2016)]

Word Piece Tokenization

1. Initialize with tokens for all characters

2. While vocabulary size is below the target size:

    1. Build a language model over the corpus (e.g., unigram language model)

    2. Merge pieces that lead to highest improvement in language model perplexity

- Need to choose a language model that will make the process tractable
- Often a unigram language model (e.g., SentencePiece library)
- Particularly suitable for machine translation

[Schuster and Nakajima (2012), Wu et al. (2016), Kudo and Richardson (2018)]

Where to get BERT?

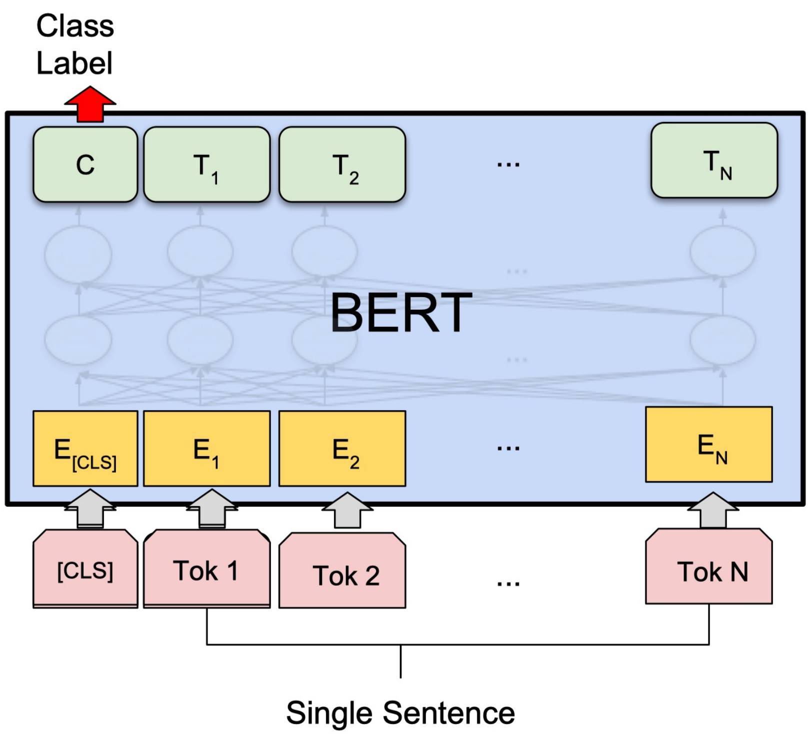
- The Transformers library: https://github.com/huggingface/transformers
- Provides state-of-the-art implementation of many models, including BERT and RoBERTa
- Including pre-trained models

Using BERT

- Use the pre-trained model as the first “layer” of your final model
- Train with fine-tuning using your supervised data
- Fine-tuning recipe: 1-3 epochs, batch size 2-32, learning rate 2e-5 - 5e-5
  - Large changes to weights in top layers (particularly in last layer to route the right information to [CLS])
  - Smaller changes to weights lower down in the transformer
  - Small learning rate and short fine-tuning schedule mean weights don’t change much
  - More complex recipes exist, but often not necessary (see Zhang et al. 2021 for study of stability and good practices)

Sentence Classification with BERT

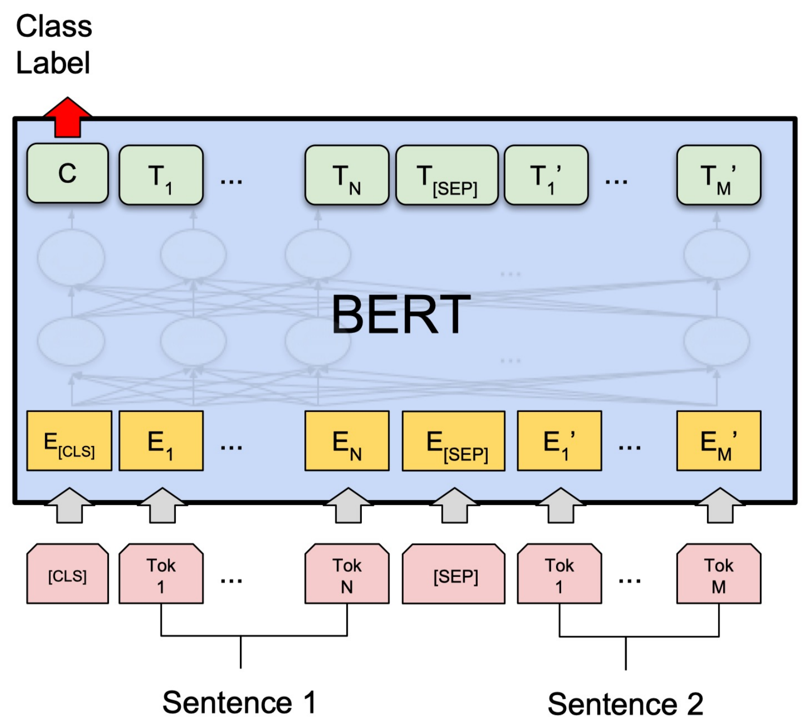
- CLS representation is used to provide classification decision
- Example tasks:
    - Sentiment classification
    - Linguistic acceptability
    - Text categorization



[figure from Devlin et al. 2018]

Sentence-pair Classification with BERT

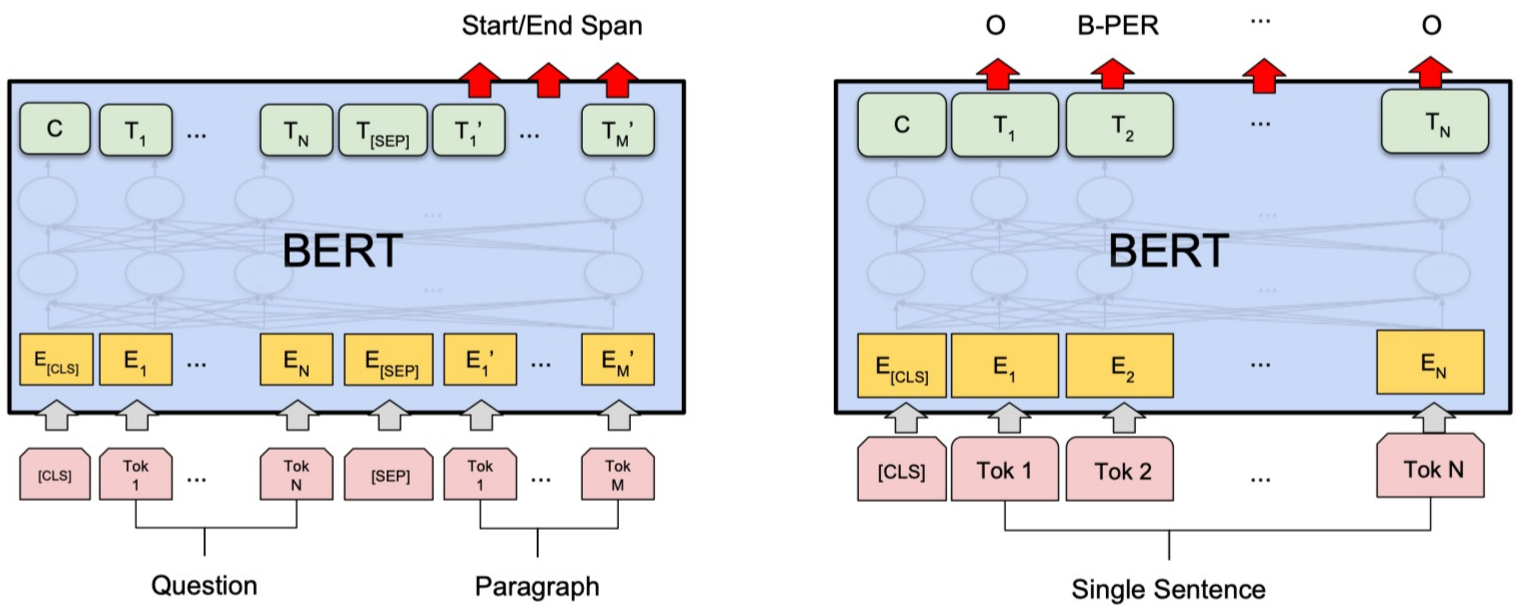
- Feed both sentences, and CLS representation used for classification
- Example tasks:
  - Textual entailment
  - Question paraphrase detection
  - Question-answering pair classification
  - Semantic textual similarity
  - Multiple choice question answering



[figure from Devlin et al. 2018]

Tagging with BERT

- Can do for a single sentence or a pair
- Tag each word piece
- Example tasks: span-based question answering, name-entity recognition, POS tagging



[figure from Devlin et al. 2018]

Results

- Fine-tuned BERT (and its variants) outperforms known methods on most NLP supervised tasks
- The larger models perform better, but even the small BERT performs better than prior methods
- Variants quickly outperformed human performance on several tasks, including span-based question answering — but what does this mean beyond the benchmarks is less clear
- Started an arms race (between industry labs) on bigger and bigger models

Hard to do with BERT

- BERT cannot generate text (at least not in an obvious way)
  - Not an autoregressive model, can do weird things like stick a [MASK] at the end of a string, fill in the mask, and repeat
- Masked language models are intended to be used primarily for "analysis" tasks

What does BERT Learn?

- A lot of recent work studying this problem
- Some very interesting results
- But, it’s not completely clear how to interpret them

What does BERT Learn?

- Try to solve different linguistic tasks given each level, without fine-tuning
- Goal: see what information each new level adds
- Method: try to solve different tasks using mixing weights on levels
- Each task classifier takes a single mixed hidden representation $\mathbf{h}_{i,\tau}$ or a pair of representations

$$i : \text{token index}$$

$$K : \text{number of levels}$$

$$\tau : \text{task}$$

$$\gamma_\tau : \text{task parameter}$$

$$\mathbf{a}_\tau : \text{mixing parameters}$$

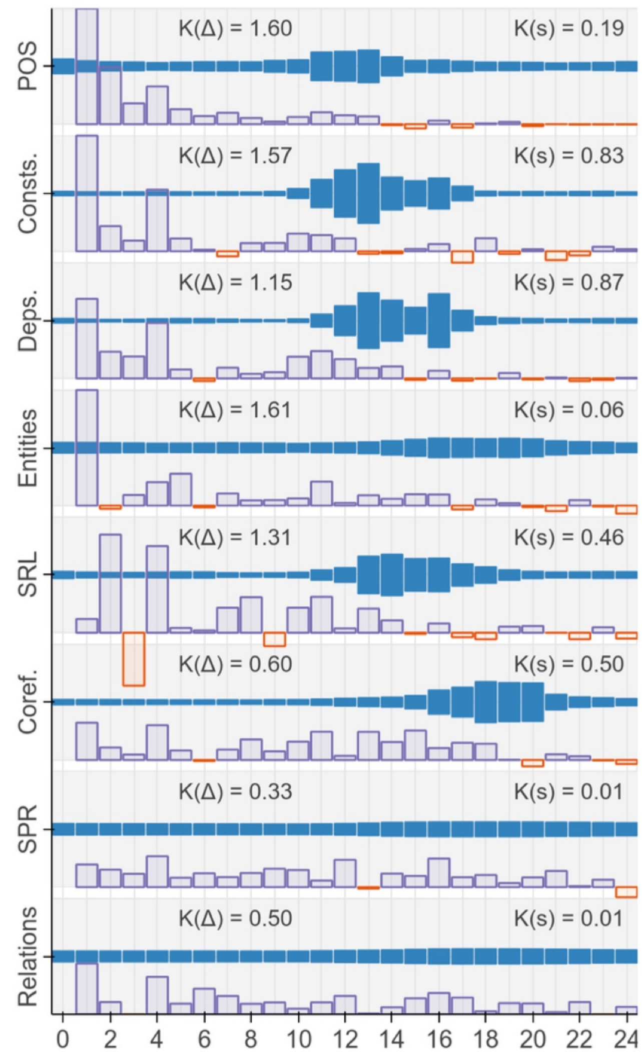
$$\mathbf{s}_\tau = \text{softmax}(\mathbf{a}_\tau)$$

$$\mathbf{h}_{i,\tau} = \gamma_\tau \sum_{k=0}^{K} s_\tau^k \mathbf{h}_i^k$$

[Tenney et al. 2019]

What does BERT Learn?

- Each plot shows a task
- Plots show s_τ^k weights magnitude in blue, and the number of self-attention levels
- The performance delta when adding this layer is in purple
- Largely: higher level semantic tasks happen in later levels



[figure from Tenney et al. (2019)]

Vision-language Reasoning

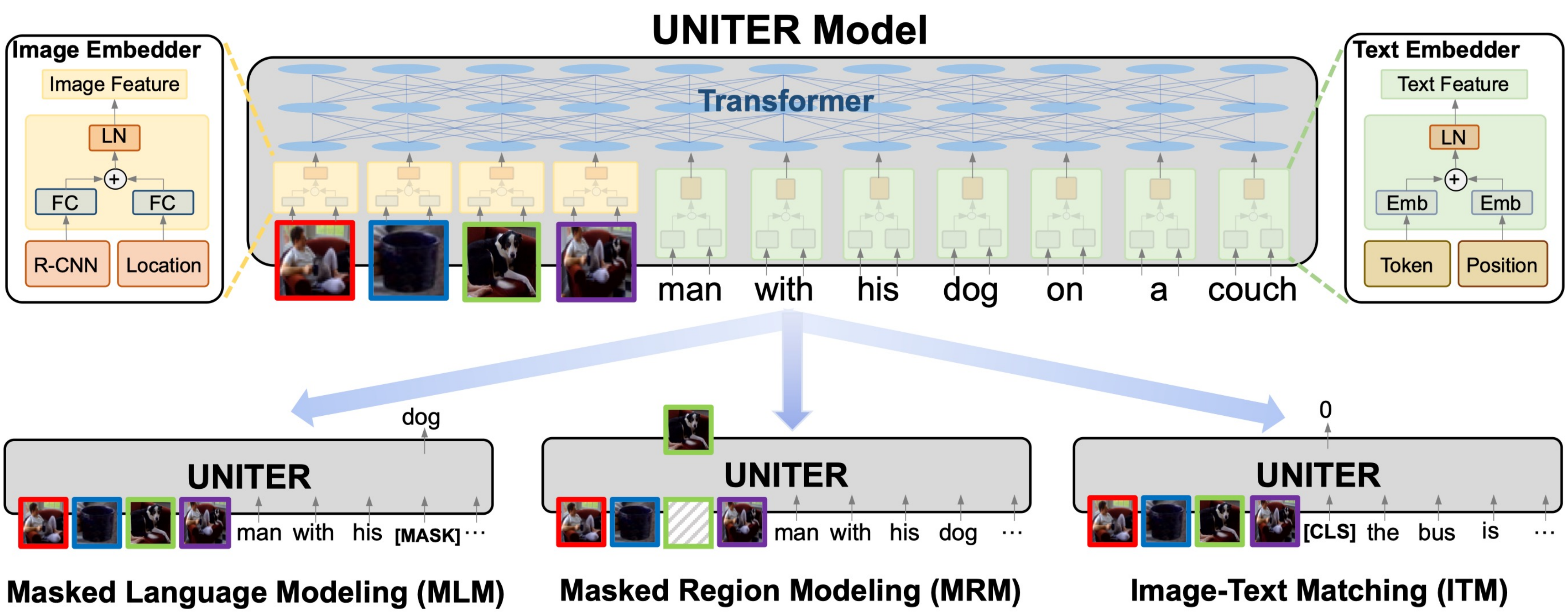
- Goal: pre-trained representations for language and vision, where the input is a sentence and image
- Self-attention in BERT allows attending between two sentences
- How can we extend that to a sentence paired with an image?

Vision-language Reasoning

- Solution: pre-process the image to extract bounding boxes around objects
- Now the image is an unordered list of discrete objects
- Objectives: masked language model + masked region modeling + image-text matching

[Li et al. 2019; Tan and Bansal 2019; Chen et al. 2019; and several other simultaneous papers]

Vision-language Reasoning



[figure from Chen et al. 2019]

Results

- Similar trend to what we observe with BERT
- State of the art on 13 vision+language benchmarks
- Similar to BERT, there larger is better

Tasks		SOTA	ViLBERT	VLBERT	Unicoder -VL	VisualBERT	LXMERT	UNITER BASE	UNITER LARGE
VQA	test-dev	70.63	70.55	70.50	-	70.80	72.42	72.27	**73.24**
	test-std	70.90	70.92	70.83	-	71.00	72.54	72.46	**73.40**
VCR	Q→A	72.60	73.30	74.00	-	71.60	-	75.00	**77.30**
	QA→R	75.70	74.60	74.80	-	73.20	-	77.20	**80.80**
	Q→AR	55.00	54.80	55.50	-	52.40	-	58.20	**62.80**
$NLVR^2$	dev	54.80	-	-	-	67.40	74.90	77.14	**78.40**
	test-P	53.50	-	-	-	67.00	74.50	77.87	**79.50**
SNLI-VE	val	71.56	-	-	-	-	-	78.56	**79.28**
	test	71.16	-	-	-	-	-	78.02	**78.98**
ZS IR (Flickr)	R@1	-	31.86	-	42.40	-	-	62.34	**65.82**
	R@5	-	61.12	-	71.80	-	-	85.62	**88.88**
	R@10	-	72.80	-	81.50	-	-	91.48	**93.52**
IR (Flickr)	R@1	48.60	58.20	-	68.30	-	-	71.50	**73.66**
	R@5	77.70	84.90	-	90.30	-	-	91.16	**93.06**
	R@10	85.20	91.52	-	94.60	-	-	95.20	**95.98**
IR (COCO)	R@1	38.60	-	-	44.50	-	-	48.42	**51.72**
	R@5	69.30	-	-	74.40	-	-	76.68	**78.41**
	R@10	80.40	-	-	84.00	-	-	85.90	**86.93**
ZS TR (Flickr)	R@1	-	-	-	61.60	-	-	75.10	**77.50**
	R@5	-	-	-	84.80	-	-	93.70	**96.30**
	R@10	-	-	-	90.10	-	-	95.50	**98.50**
TR (Flickr)	R@1	67.90	-	-	82.30	-	-	84.70	**88.20**
	R@5	90.30	-	-	95.10	-	-	97.10	**98.40**
	R@10	95.80	-	-	97.80	-	-	99.00	**99.00**
TR (COCO)	R@1	50.40	-	-	59.60	-	-	63.28	**66.60**
	R@5	82.20	-	-	85.10	-	-	87.04	**89.42**
	R@10	90.00	-	-	91.80	-	-	93.08	**94.26**
Ref-COCO	val	87.51		-	-	-	-	91.64	**91.84**
	testA	89.02	-	-	-	-	-	92.26	**92.65**
	testB	87.05	-	-	-	-	-	90.46	**91.19**
	val^d	77.48	-	-	-	-	-	81.24	**81.41**
	$testA^d$	83.37	-	-	-	-	-	86.48	**87.04**
	$testB^d$	70.32	-	-	-	-	-	73.94	**74.17**
Ref-COCO+	val	75.38	-	78.44	-	-	-	82.84	**84.04**
	testA	80.04	-	81.30	-	-	-	85.70	**85.87**
	testB	69.30	-	71.18	-	-	-	78.11	**78.89**
	val^d	68.19	72.34	71.84	-	-	-	74.72	**74.94**
	$testA^d$	75.97	78.52	77.59	-	-	-	80.65	**81.37**
	$testB^d$	57.52	62.61	60.57	-	-	-	65.15	**65.35**
Ref-COCOg	val	81.76	-	-	-	-	-	86.52	**87.85**
	test	81.75	-	-	-	-	-	86.52	**87.73**
	val^d	68.22	-	-	-	-	-	74.31	**74.86**
	$test^d$	69.46	-	-	-	-	-	74.51	**75.77**

Text Generation Evaluation with BERT

- Can we use BERT to evaluate language generation? Such as MT, paraphrase, caption generation, etc.
- Input is two sentences: a reference and a system output
- Output: a score that tells us how similar they are

Text Generation Evaluation with BERT

- How do we do it usually? Bleu

- Bleu matches n-grams between the reference and the candidate

- When does this fail?

Reference
the weather is cold today

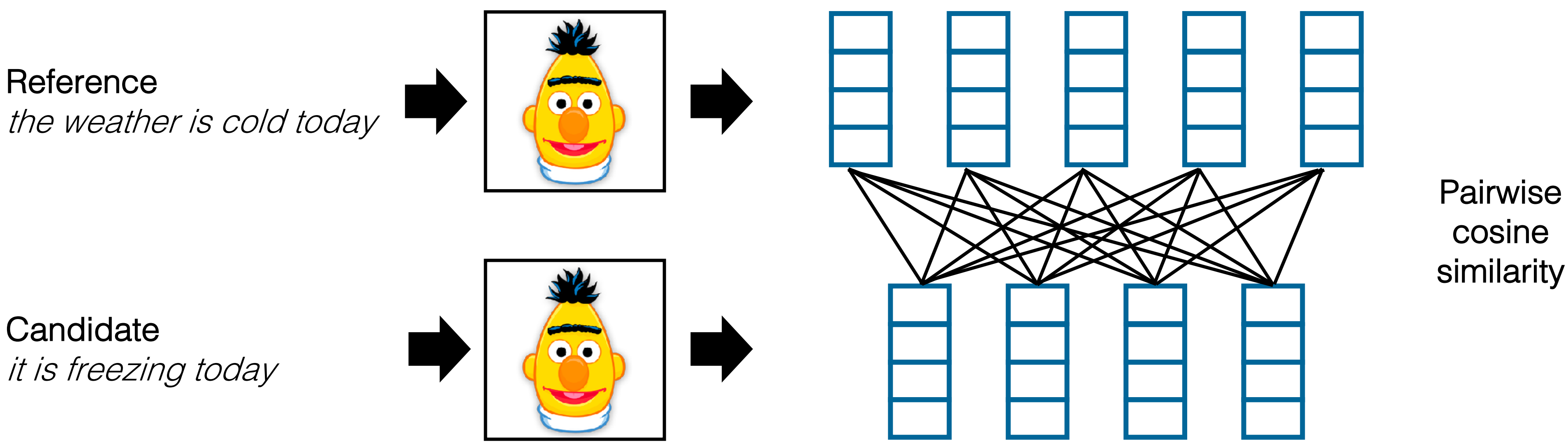
Candidate
it is freezing today

Text Generation Evaluation with BERT

- How do we do it usually? Bleu
- Bleu matches n-grams between the reference and the candidate
- When does this fail?
- Sensitive to exact phrasing and word choices
- This can bring about false negatives

BERTScore

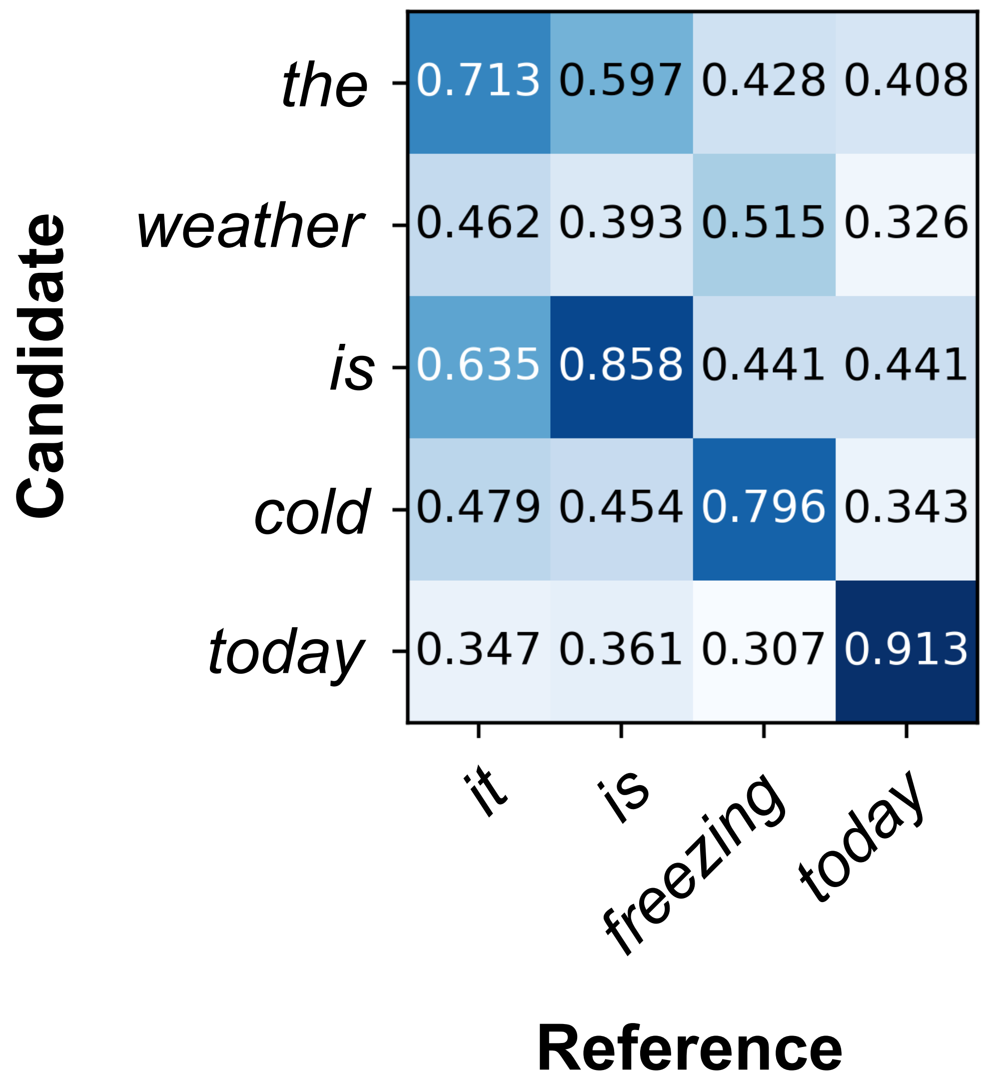
- Instead of string matching, like in Bleu
- Use BERT embedding to compute similarity



Matching

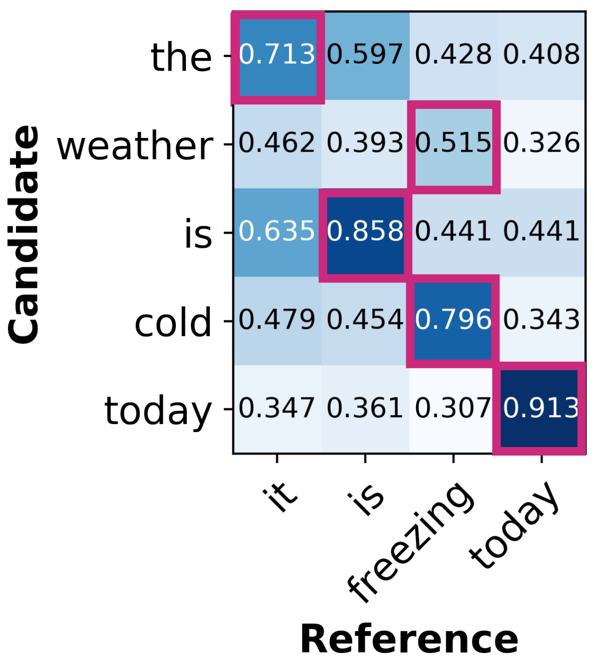
- Compute similarity between all possible pairs

- Build a similarity matrix



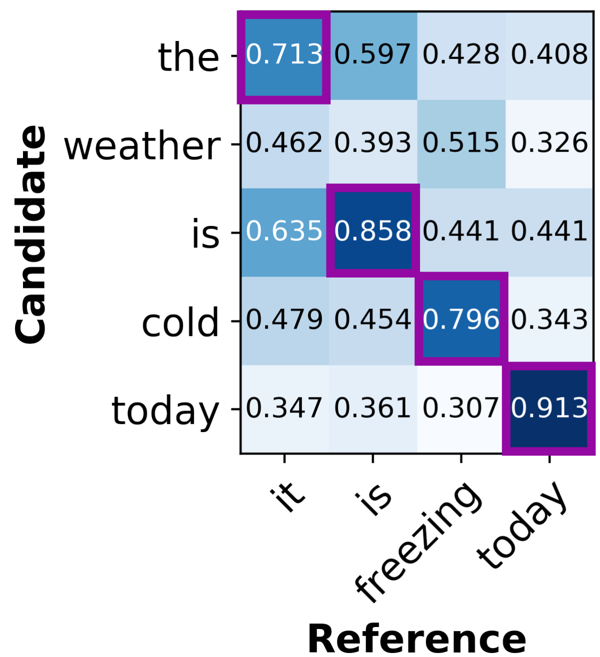
Greedy Matching

Precision



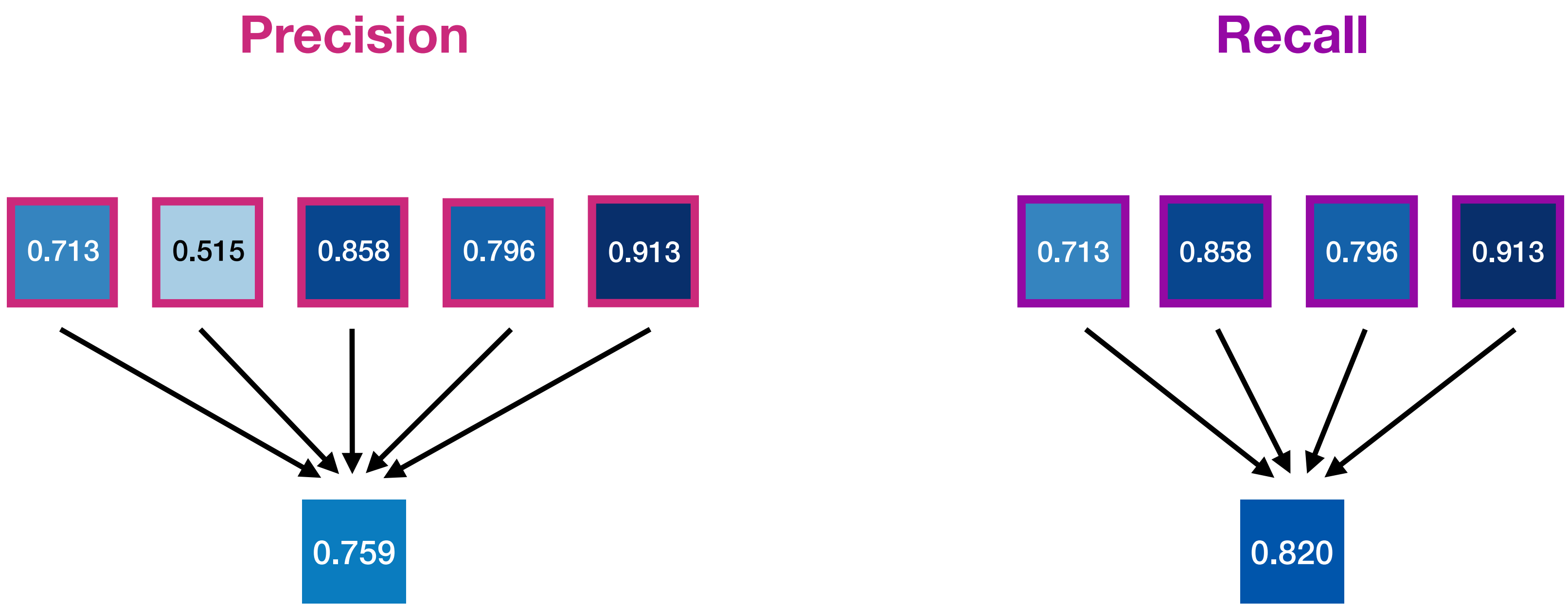
Match words in candidate to reference

Recall



Match words in reference to candidate

Greedy Matching: Aggregate



$$\mathrm{F1} = 2\,\frac{\mathrm{Precision}\cdot\mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}}$$

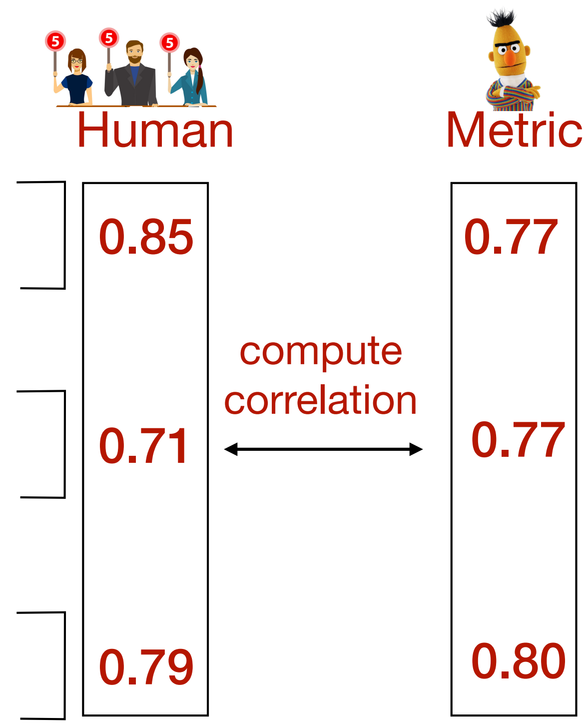
Evaluating Evaluation

- Collect human judgements
- Measure correlations with your metric

Reference: *The weather is cold today.*
Candidate: *It is freezing today.*

Reference: *The garden is nice.*
Candidate: *The garden was pretty.*

Reference: *I like apples very much.*
Candidate: *I love apples.*



Evaluating Evaluation