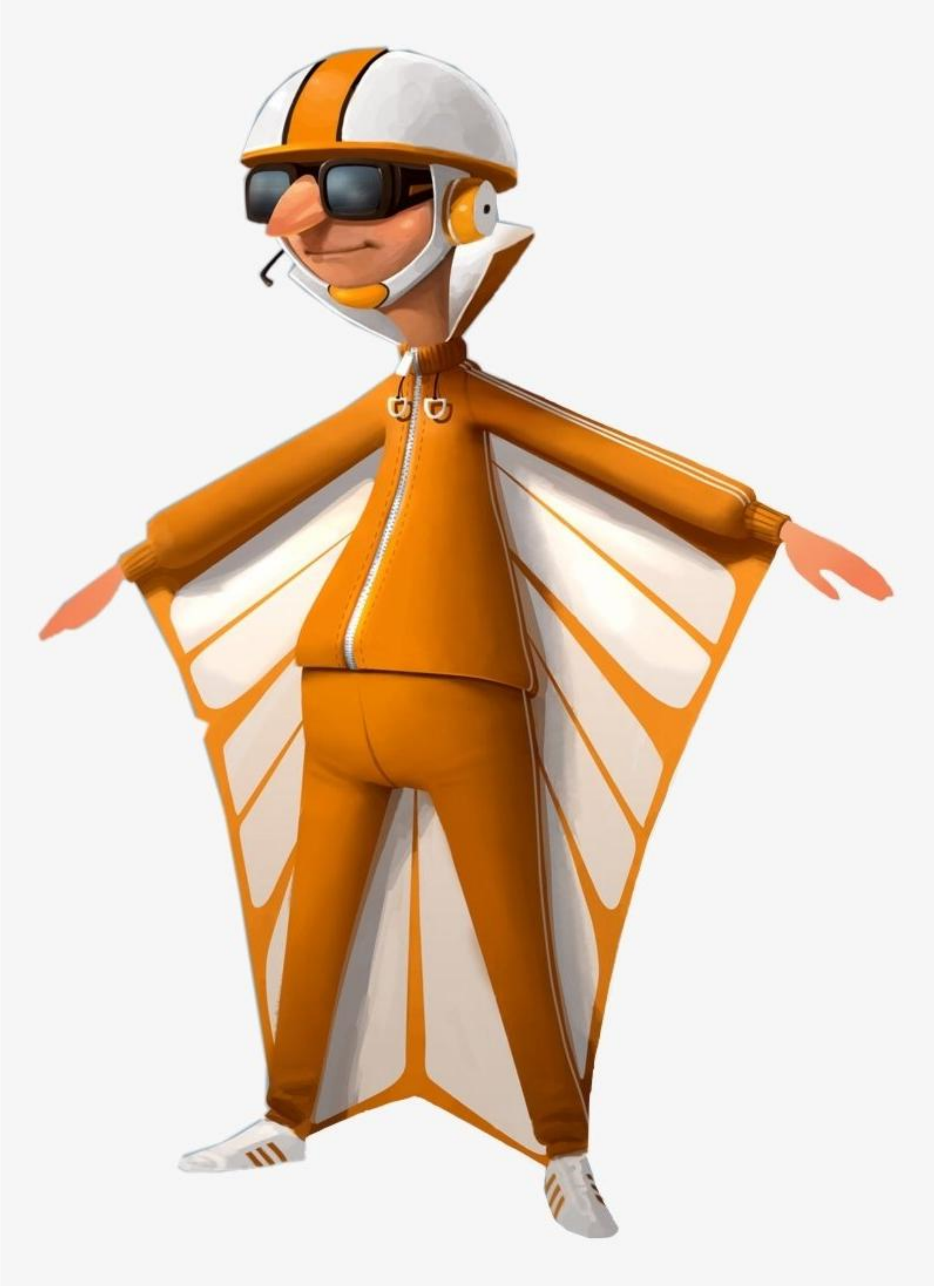
Vector Semantics II

CSC485

Lecture 11



Terminology Hell

- "Embedding"
  - Embedding layer: `torch.nn.Embedding`
    - Linear layer: one hot index -> vectorized representation
    - Basically, a big look up table
  - Vector(ized) Representation
    - Using an n-dim vector to represent a word. The vector.
  - Hidden Representation; Hidden State
    - The intermediate output of a neural network
    - Neural LM: use this as the vectorized representation
  - Word Embedding:
    - The model/system/algorithm that generate a vectorized representation given a word.
  - Word Embedding:
    - The generated vectorized representation.

adverbs	verbs	adjectives	nouns
appropriately	actualize	24/7	action items
assertively	administrate	24/365	adoption
authoritatively	aggregate	accurate	alignments
collaboratively	architect	adaptive	applications
compellingly	benchmark	agile	architectures
competently	brand	alternative	bandwidth
completely	build	an expanded array	benefits
continually	cloudify	of	best practices
conveniently	communicate	B2B	catalysts for change
credibly	conceptualize	B2C	channels
distinctively	coordinate	backend	clouds
dramatically	create	backward-	collaboration and idea-
dynamically	cultivate	compatible	sharing
efficiently	customize	best-of-breed	communities
energistically	deliver	bleeding-edge	content
enthusiastically	deploy	bricks-and-clicks	convergence
fungibly	develop	business	core competencies
globally	dinintermediate	clicks-and-mortar	customer service
holisticly	disseminate	client-based	data

[The Corporate B.S. Generator]

WSD!

Contextual vs. Global Word Embedding

- Global Word Embedding
  - One vector representation word-type
  - word2vec, GloVe
- Contextual Word Embedding
  - One vector representation word-token
  - RNN, LSTM, BERT, GPT…

Lecture 3: Primitives: lexical categories or parts of speech.

- Each word-type is a member of one or more.
- Each word-token is an instance of exactly one.

The Language Modelling Pipeline

- Collect large quantity of unstructured data
  - Wikipedia articles, social media post, news articles…
  - Famous open-source: WikiText-2/103 (100M Tokens), Dolma (3T tokens)
- Tokenization
  - `The University of Toronto (UToronto or U of T) is a public research university in Toronto, Ontario, Canada, located on the grounds that surround Queen's Park.`
  - `['The', 'ĠUniversity', 'Ġof', 'ĠToronto', 'Ġ(', 'UT', 'or', 'onto', 'Ġor', 'ĠU', 'Ġof', 'ĠT', ')', 'Ġis', 'Ġa', 'Ġpublic', 'Ġresearch', 'Ġuniversity', 'Ġin', 'ĠToronto', ',', 'ĠOntario', ',', 'ĠCanada', ',', 'Ġlocated', 'Ġon', 'Ġthe', 'Ġgrounds', 'Ġthat', 'Ġsurround', 'ĠQueen', "'s", 'ĠPark', '.’]`
- Perform Language Modelling Task:
  - Next Word Prediction, Masked Language Modelling, …

Tokenization and Tokenizers

- Character-level language modeling:
    - Classifying Names with a Character-Level RNN [Pytorch Tutorial]
    - Good with Chinese
    - Other languages: inefficient use of data
- Tokenization: breaks down text into smaller units, often called tokens.
    - `text.split()`
- The only difficulty: unknown token.
    - Special <unk> token

#longexposurephotography	Rechtsschutzversicherungsgesellschaft
Long exposure photography	Rechts Schutz Versicherung s Gesellschaft
	legal protection insurance company

Tokenization and Tokenizers

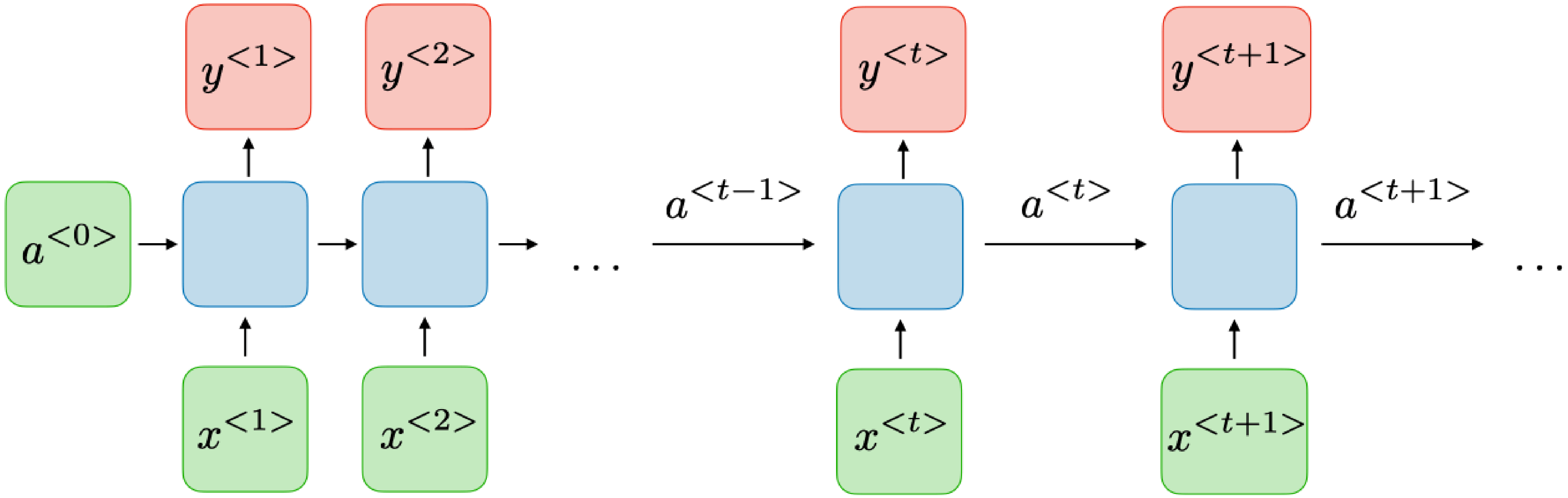
- Solution: **Break a word down into word pieces!**
- Slightly different encoding styles

  colorless green ideas sleep furiously

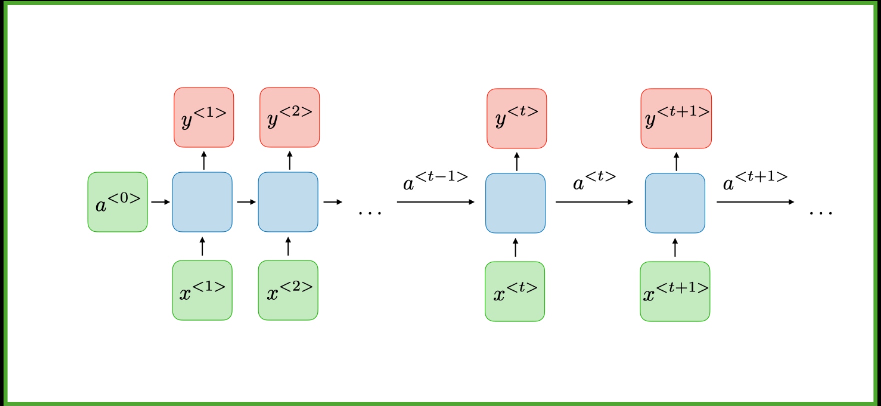
  - BERT: 'color', '##less', 'green', 'ideas', 'sleep', 'furiously’
  - GPT/LLaMA: 'color', 'less', 'Ġgreen', 'Ġideas', 'Ġsleep', 'Ġfuriously’
  - XLM: 'color', 'less</w>', 'green</w>', 'ideas</w>', 'sleep</w>', 'furiously</w>’
- Algorithms: see [this tutorial]
  - Train with a large corpus
  - Byte pair encoding (BPE):
    - break everything down into characters
    - merge the most frequent pairs
    - repeat until vocab size reached.
  - Wordpiece:
    - Score = (freq_of_pair)/(freq_of_first_element × freq_of_second_element)

Contextual Word Embedding

- Recurrent neural network (RNN)



```python
class RNN(nn.Module):

    def __init__(self, input_size, hidden_size, output_size):
        # i: input token, h: hidden state, o: output
        self.i2h = nn.Embedding(input_size, hidden_size)
        self.h2h = nn.Linear(hidden_size, hidden_size)
        self.h2o = nn.Linear(hidden_size, output_size) # output_size: number of labels

    def forward(self, x, hidden_state):
        x = self.i2h(x)
        hidden_state = self.h2h(hidden_state)
        hidden_state = torch.tanh(x + hidden_state)
        out = self.h2o(hidden_state)
        return out, hidden_state
```

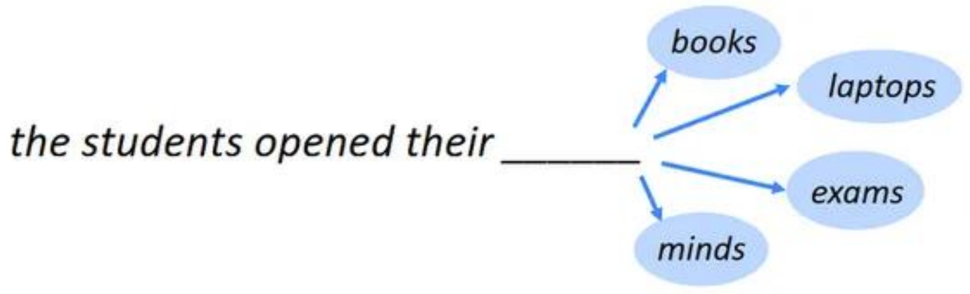


Recall: Language Modelling Task

- Final goal: predict/estimate the probability of a sequence

Probability(*Some sentence over here.*)

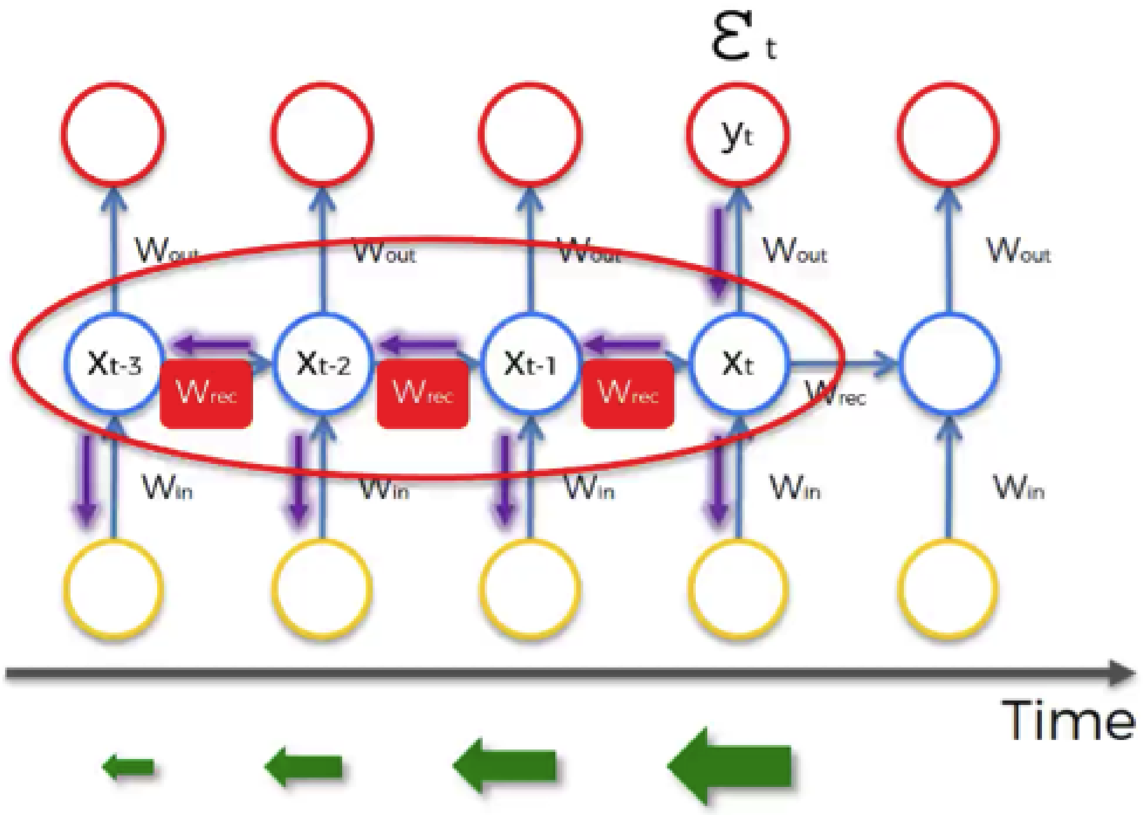
- Actual task:
  - Predict the next word
  - MLM



- In a perfect world:
  - The RNN hidden states should be able capture all contextual information

Reality

The Vanishing Gradient Problem



$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \leq t \leq T} \frac{\partial \mathcal{E}_t}{\partial \theta} \qquad (3)$$

$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \leq k \leq t} \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \right) \qquad (4)$$

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \geq i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \geq i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1})) \qquad (5)$$

W_{rec} ~ small	⇨	Vanishing
W_{rec} ~ large	⇨	Exploding

Formula Source: Razvan Pascanu et al. (2013)

Also, we need more! What of larger semantic units?

- How can we know when larger units are similar in meaning?
  - *CTV News*: Poilievre-led attempt to bring down Trudeau minority over carbon tax fails.
  - *CBC News*: Liberals survive non-confidence vote on carbon tax with Bloc, NDP backing.
  - The Beaverton: Co-worker that everyone hates surprised he can't get colleagues to do what he wants.

NATIONAL - 2 WEEKS AGO

Co-worker that everyone hates surprised he can't get colleagues to do what he wants

OTTAWA – Local man Pierre Poilievre, an employee at an Ottawa small business named the House of Commons, was surprised that none of the colleagues who despise him were willing to support hi...

SHARE

POLITICS - AUGUST 18, 2020

Gender-balanced cabinet forces woman to work two jobs

OTTAWA – A gender-balanced cabinet is now requiring a woman to work two jobs after a male employee left his post for higher pursuits. Chrystia Freeland, the Deputy Prime Minister, has been ...

SHARE

RNN & next word prediction: Not good compositional representation

- Next word prediction:

$$P(t_i|t_1, t_2, \dots t_{i-1})$$

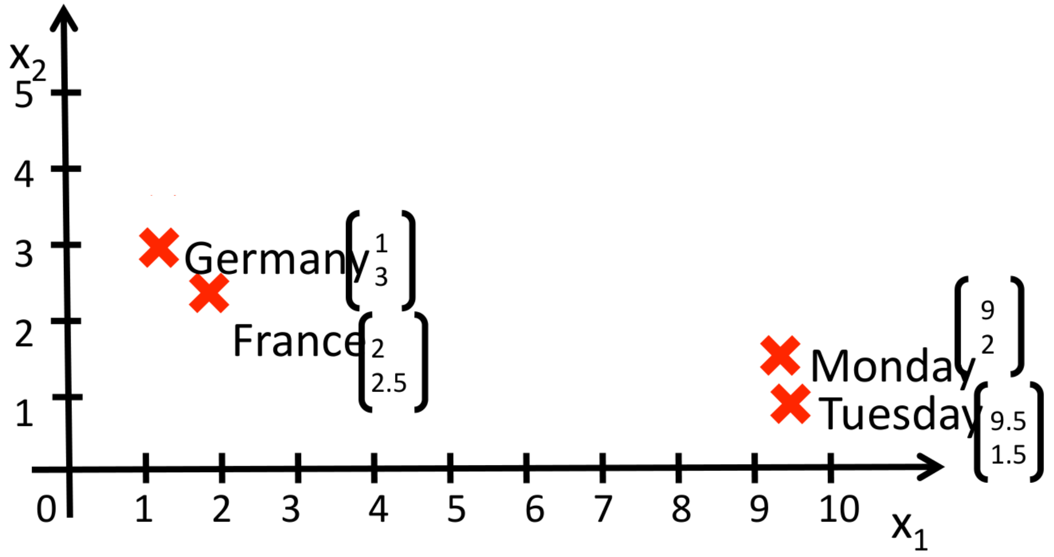
- The hidden state i is encoding information of everything from the beginning (index 0) to the very end (index i).
- We want some bigger semantic units
  - Poilievre-led attempt to **bring down Trudeau minority over carbon tax** fails.
- Some hacks may work, but not really

Also, we need more! What of larger semantic units?

- How can we know when larger units are similar in meaning?
  - *CTV News*: Poilievre-led attempt to bring down Trudeau minority over carbon tax fails.
  - *CBC News*: Liberals survive non-confidence vote on carbon tax with Bloc, NDP backing.
  - The Beaverton: Co-worker that everyone hates surprised he can't get colleagues to do what he wants.

People interpret the meaning of larger text units
– entities, descriptive terms, facts, arguments, stories –
by **semantic composition** of smaller elements.

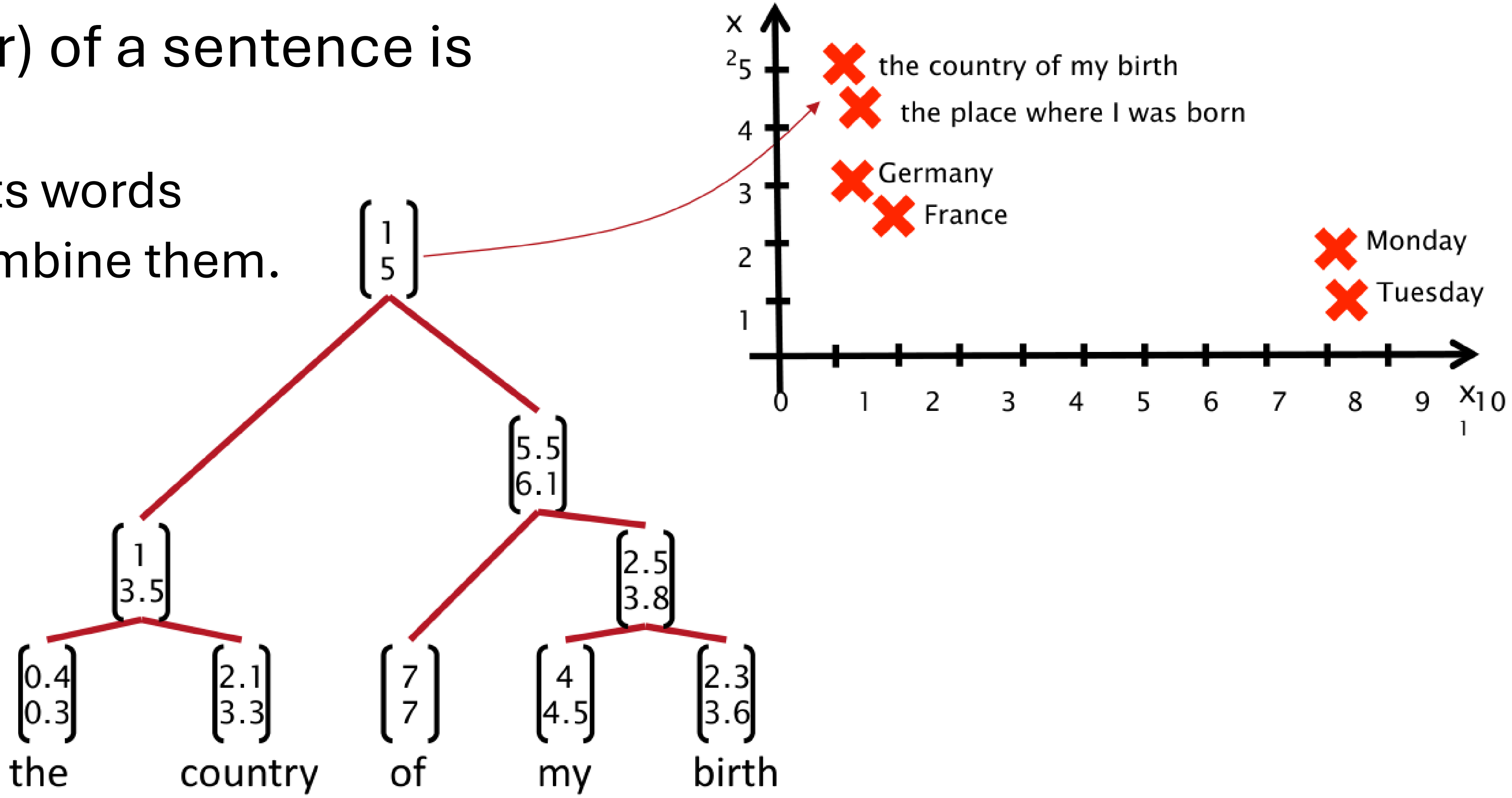
Representing Phrases as Vectors

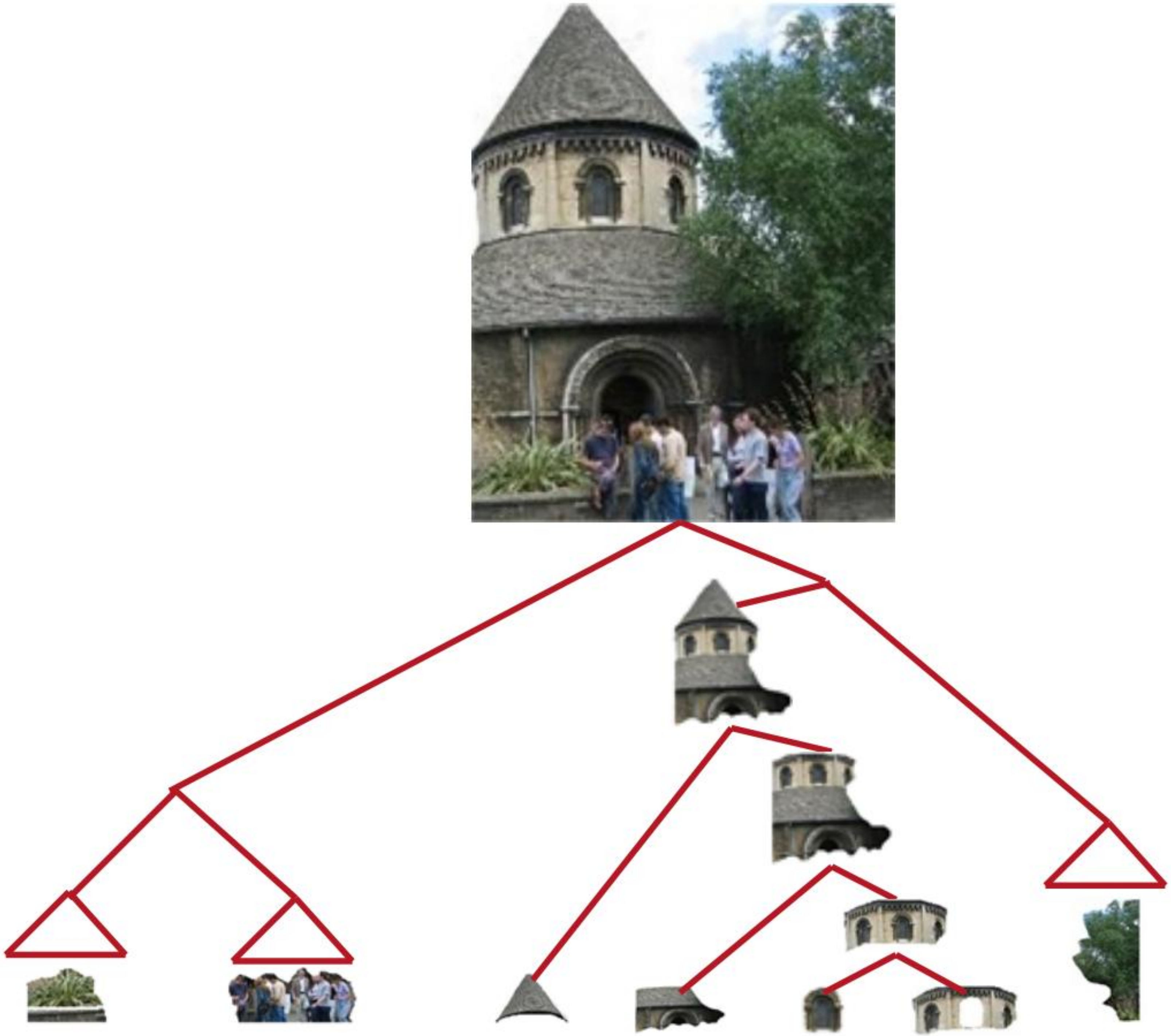


- Vector for single words are useful as features but limited.
  - the country of my birth
  - the place where I was born
- Can we extend the ideas of word vector spaces to phrases?

Understand Larger Semantic Units

- Use the principle of compositionality!
- The meaning (vector) of a sentence is determined by:
  1. the meanings of its words
  2. a method that combine them.

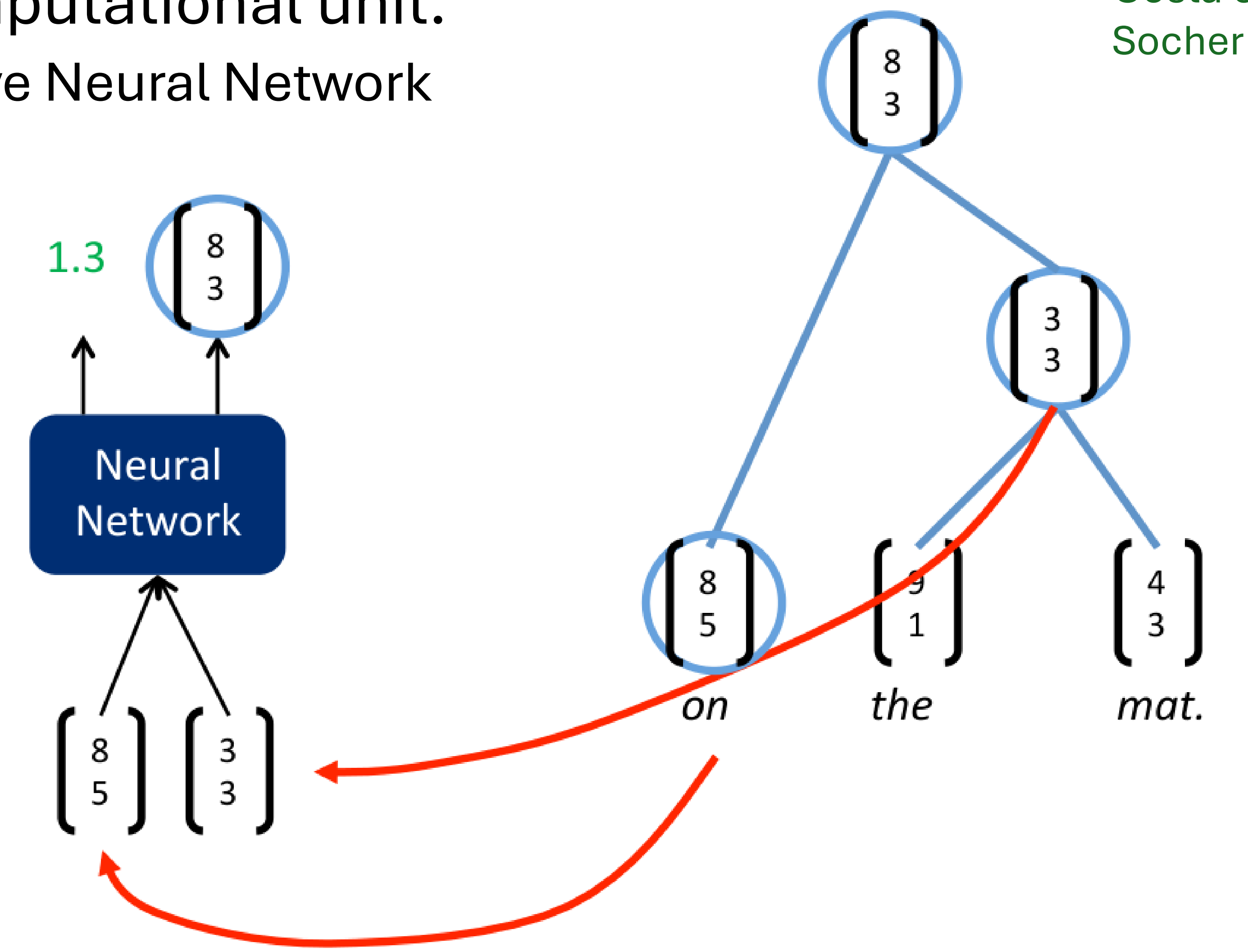




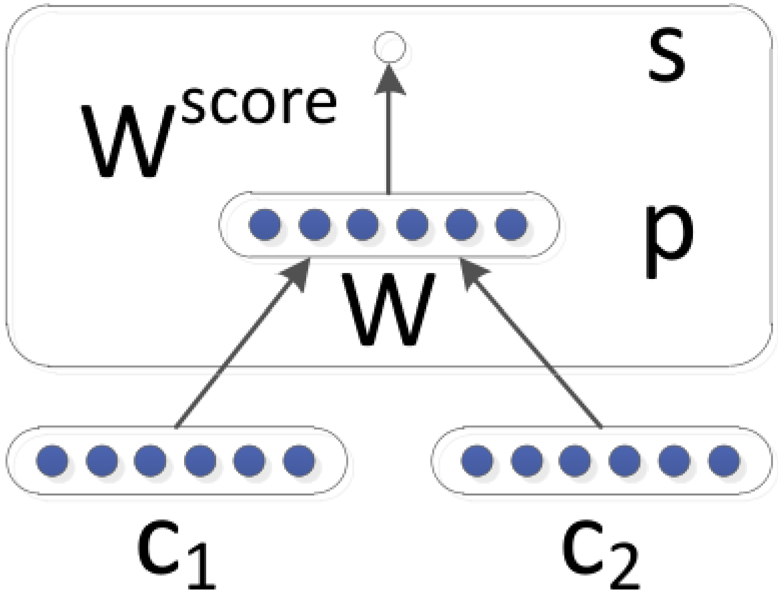
Tree RNNs

- Basic computational unit:
  - Recursive Neural Network

Goller & Küchler 1996,
Costa et al. 2003,
Socher et al. ICML, 2011.



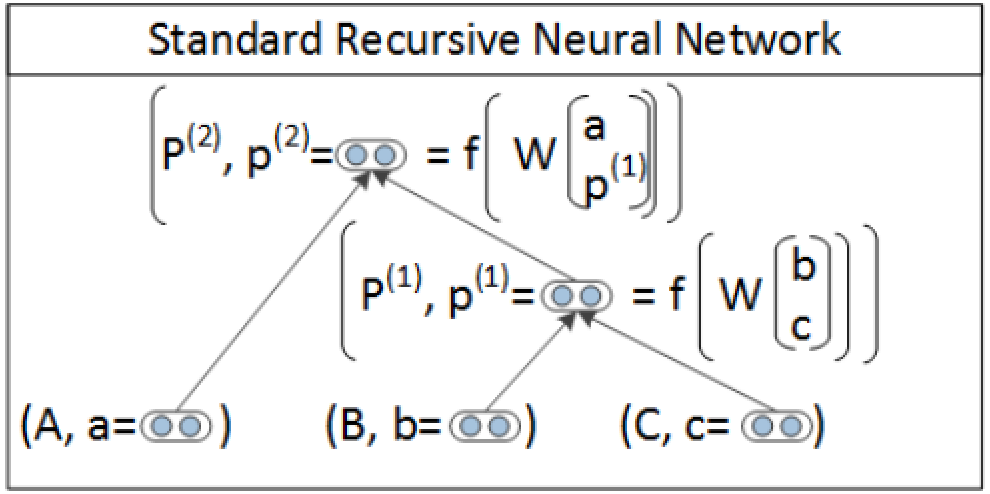
Version 1: Simple concatenation Tree RNN

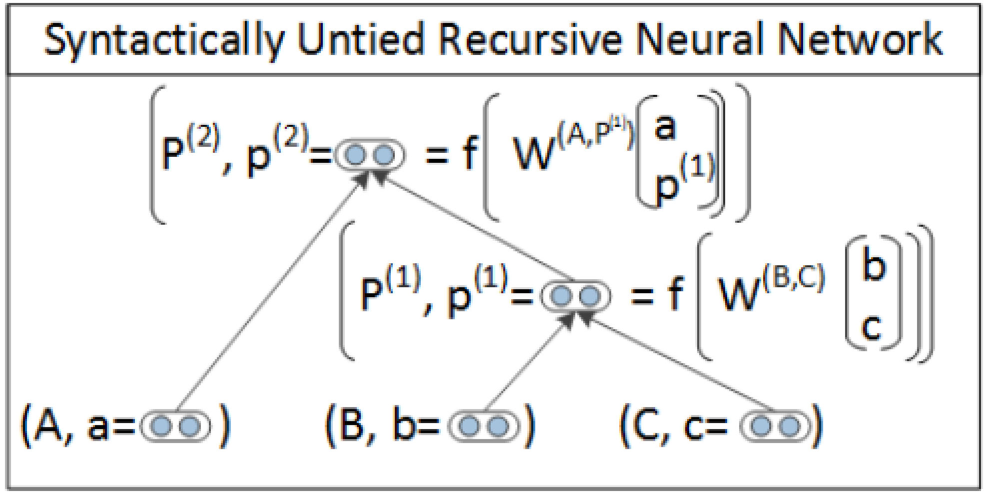


- Only a single weight matrix = composition function!
- No real interaction between the input words!
- Not adequate for human language composition function

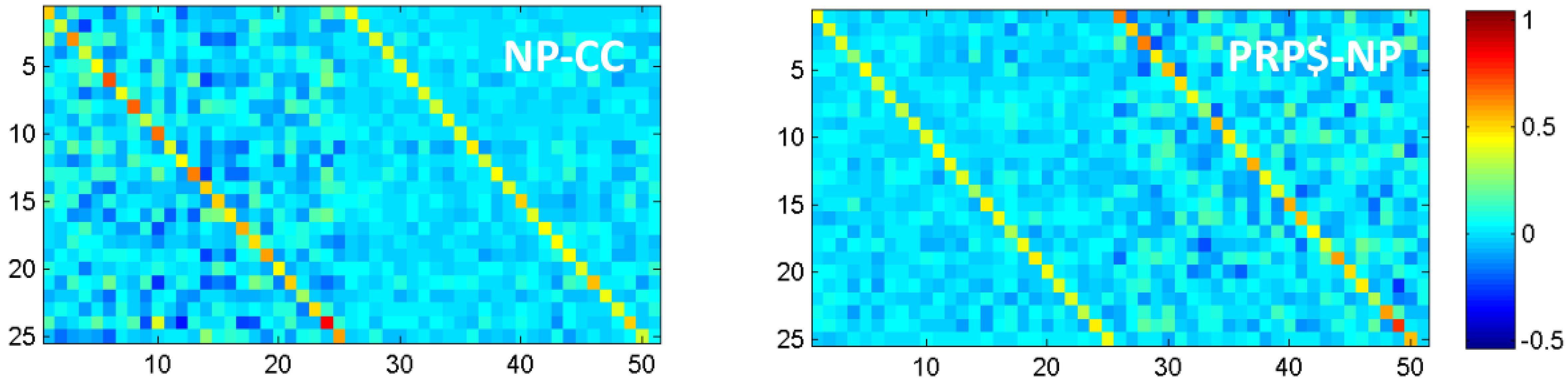
Version 2: PCFG + Syntactically-United RNN

- Idea: Condition the composition function on the syntactic categories, “untie the weights.”
- Allows for different composition functions for pairs of syntactic categories, e.g. Adv + AdjP, VP + NP.
- Combines discrete syntactic categories with continuous semantic information.





SU-RNN: Learns a soft version of head words



Head words get bigger weights in the matrices

More versions!

- Version 4: Recursive Neural Tensor Network
- Version 5: Tree-Structured Long Short-Term Memory Networks
- …

Natural Language Inference

- Can we tell if one piece of text follows from another?
  - Poilievre-led attempt to bring down Trudeau minority over carbon tax fails.
  - Liberals survive non-confidence vote on carbon tax with Bloc, NDP backing.
- Natural Language Inference = Recognizing Textual Entailment [Dagan 2005, MacCartney & Manning, 2009]

NLI: The Task

James Byron Dean refused to move without blue jeans

{**entails**, contradicts, neither}

James Dean didn’t dance without pants

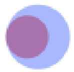
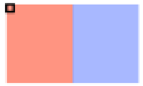
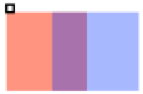
NLI: The Task

Simple task to define, but engages the full complexity of compositional semantics:

- Lexical entailment
- Quantification
- Coreference
- Lexical/scope ambiguity
- Commonsense knowledge
- Propositional atitudes
- Modality
- Factivity and implicativity
- …

Natural logic: relations

- Seven possible relations between phrases/sentences:

	$x \equiv y$	equivalence	*couch* ≡ *sofa*
	$x \sqsubset y$	forward entailment (strict)	*crow* ⊏ *bird*
	$x \sqsupset y$	reverse entailment (strict)	*European* ⊐ *French*
	$x \wedge y$	negation (exhaustive exclusion)	*human* ^ *nonhuman*
	$x \mid y$	alternation (non-exhaustive exclusion)	*cat* | *dog*
	$x \smile y$	cover (exhaustive non-exclusion)	*animal* ‿ *nonhuman*
	$x \# y$	independence	*hungry* # *hippo*

Natural logic: relation joins

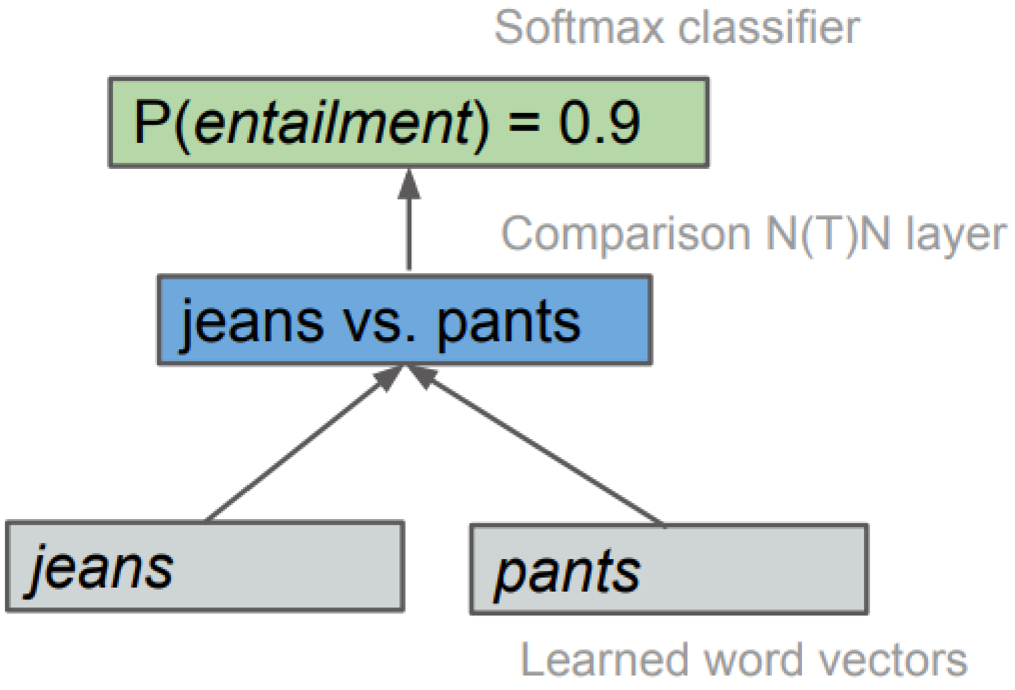
	$\equiv$	$\sqsubset$	$\sqsupset$	$\wedge$	$\mid$	$\smile$	$\#$
$\equiv$	$\equiv$	$\sqsubset$	$\sqsupset$	$\wedge$	$\mid$	$\smile$	$\#$
$\sqsubset$	$\sqsubset$	$\sqsubset$	$\cdot$	$\mid$	$\mid$	$\cdot$	$\cdot$
$\sqsupset$	$\sqsupset$	$\cdot$	$\sqsupset$	$\smile$	$\cdot$	$\smile$	$\cdot$
$\wedge$	$\wedge$	$\smile$	$\mid$	$\equiv$	$\sqsupset$	$\sqsubset$	$\#$
$\mid$	$\mid$	$\cdot$	$\mid$	$\sqsubset$	$\cdot$	$\sqsubset$	$\cdot$
$\smile$	$\smile$	$\smile$	$\cdot$	$\sqsupset$	$\sqsupset$	$\cdot$	$\cdot$
$\#$	$\#$	$\cdot$	$\cdot$	$\#$	$\cdot$	$\cdot$	$\cdot$

Can our NNs learn to make these inferences
over pairs of embedding vectors?

A minimal NN for lexical relations

[Bowman 2014]

- Words are learned embedding vectors.
- One plain TreeRNN or TreeRNTN layer
- Softmax emits relation labels
- Learn everything with SGD.



Lexical relations: results

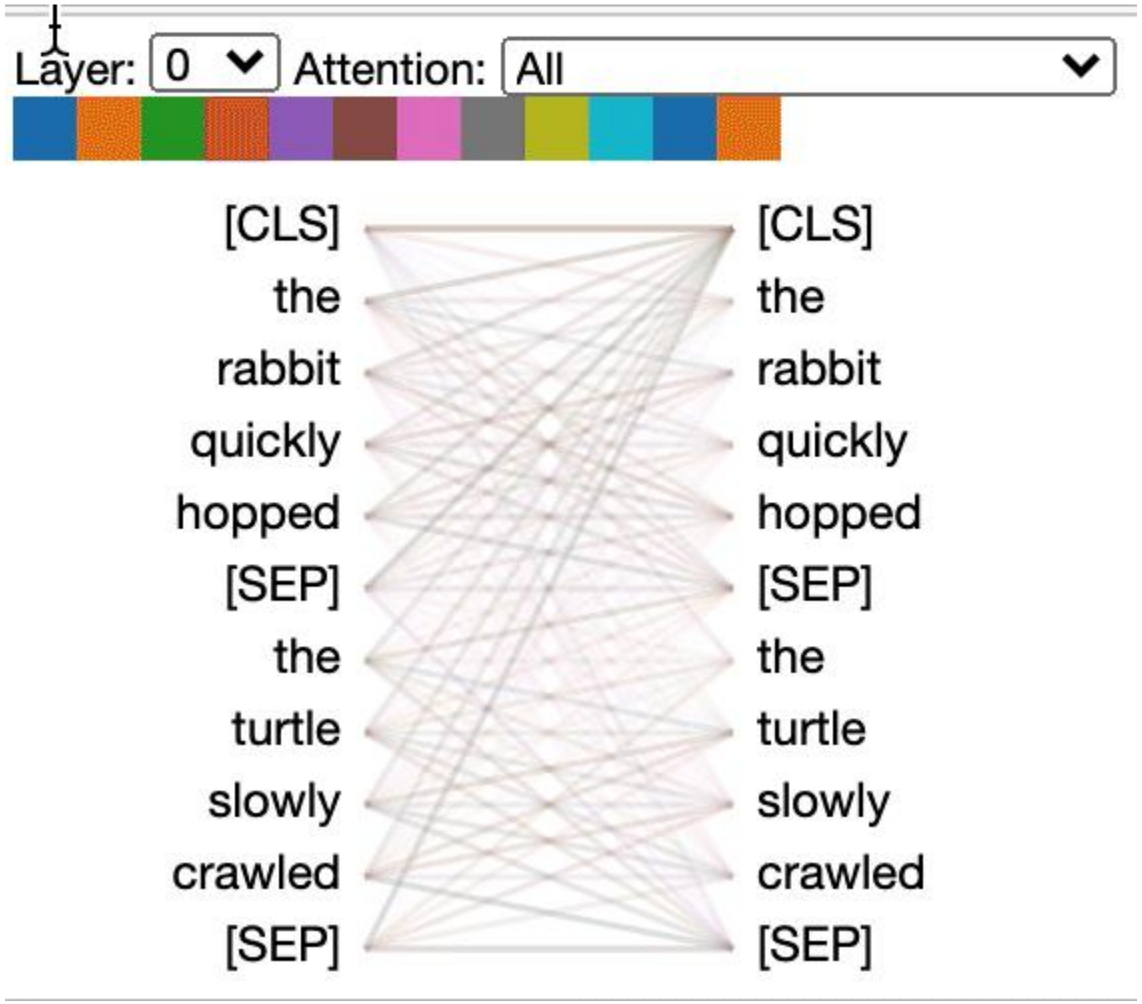
	Train	Test
# only	53.8 (10.5)	53.8 (10.5)
15d NN	99.8 (99.0)	94.0 (87.0)
15d NTN	**100 (100)**	**99.6 (95.5)**

- Both models tuned, then trained to convergence on five randomly generated datasets
- Reported figures: % correct (macroaveraged F1)
- Both NNs used 15d embeddings, 75d comparison layer

Transformers

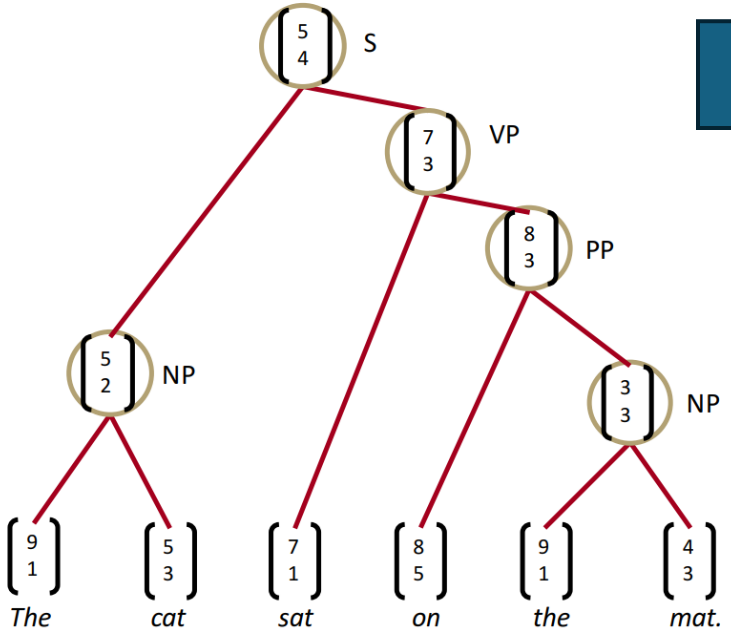
Transformer

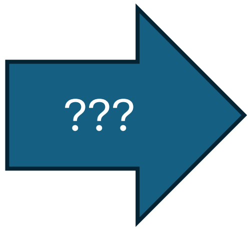
- Attention is All You Need
- Key, Query, Value… from Lecture 2
- Each token’s representation: weighted sum of other token’s representation.
- Soft “syntactic” structure!

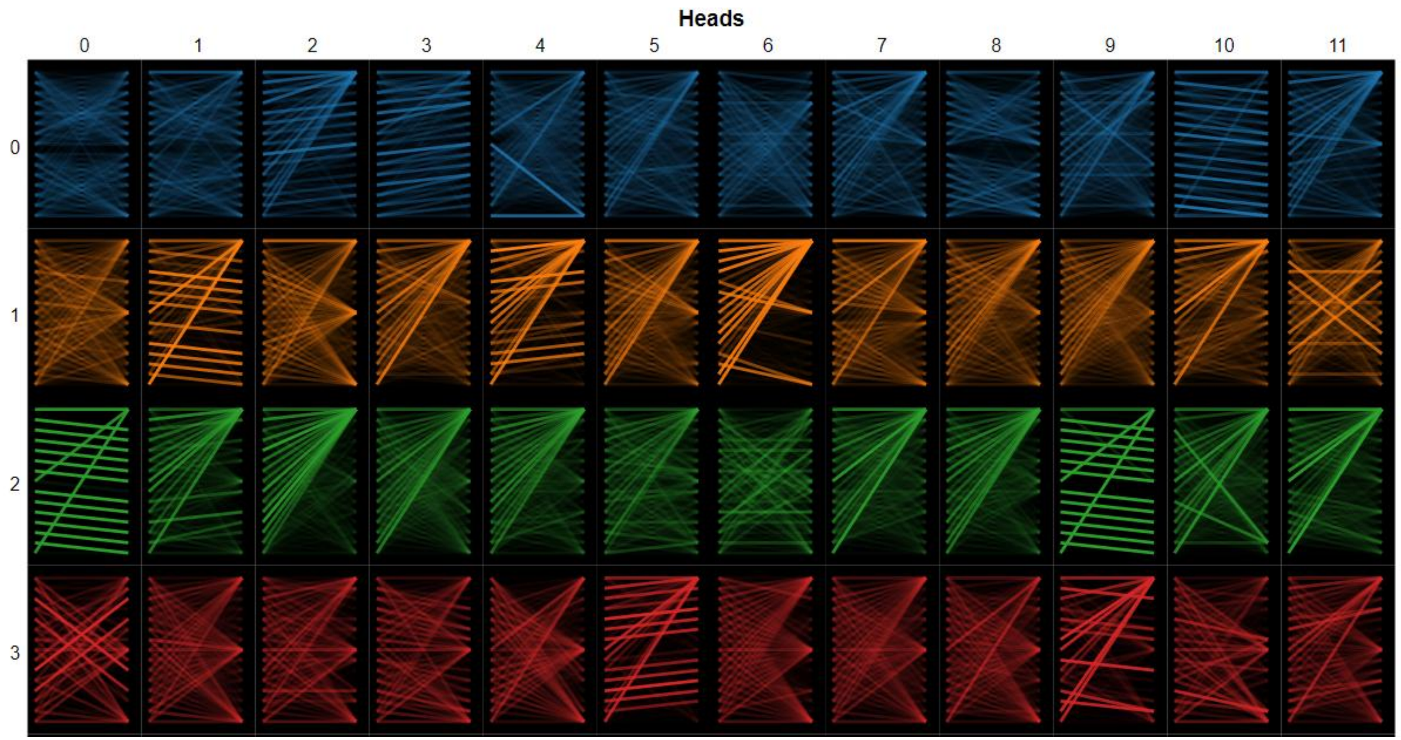


BertViz demo

Soft Syntactic Structure



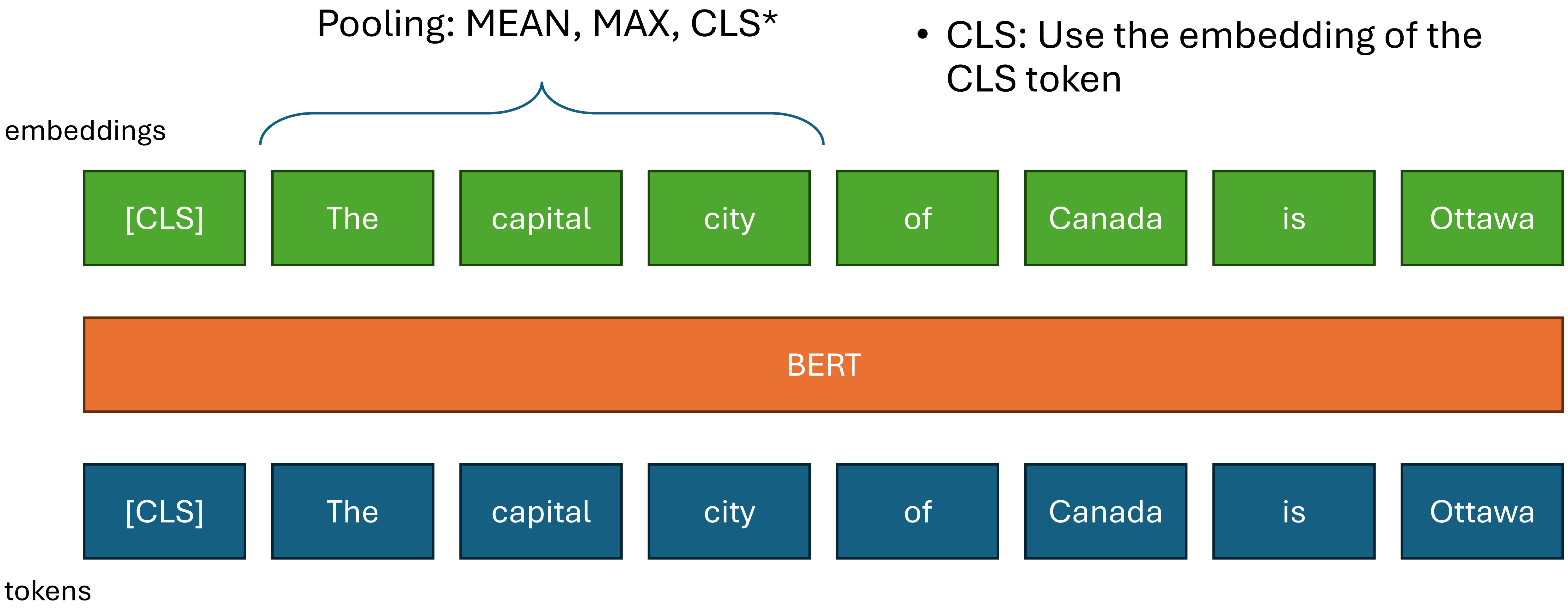




More when we reach interpretability:
Spoil alert: Transformers learn some soft syntactic structure, but nothing like formal, human syntax as we understood.

Sentence Embedding

- MEAN: take the average of all word embeddings
- MAX: take the maximum value along every dimension
- CLS: Use the embedding of the CLS token



Sentence-BERT

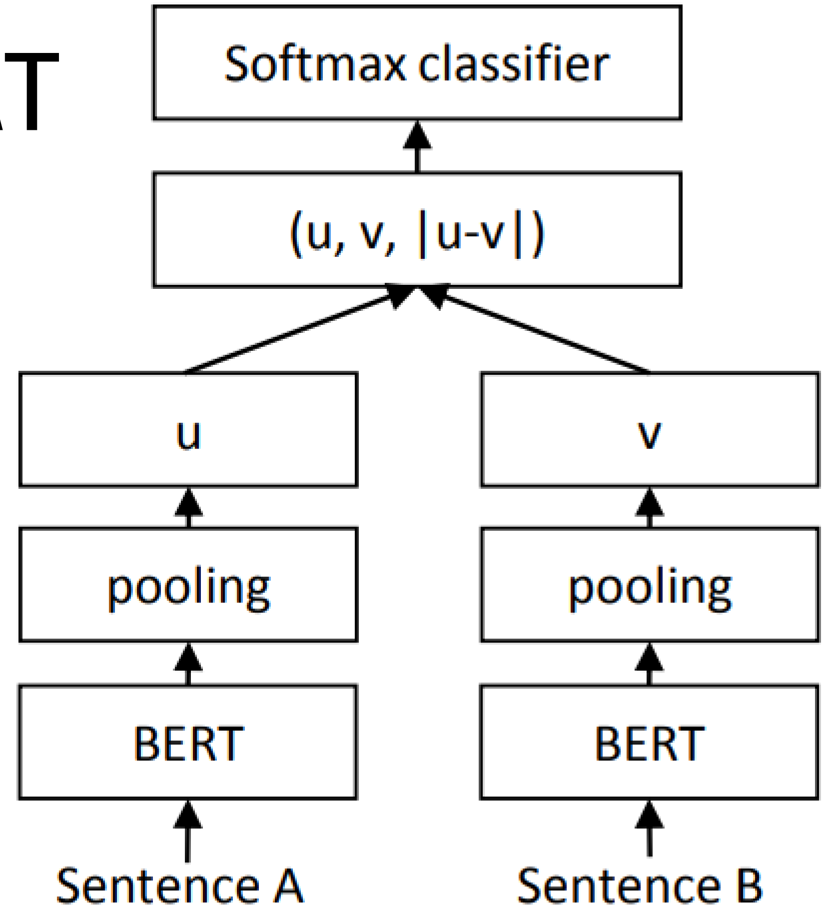
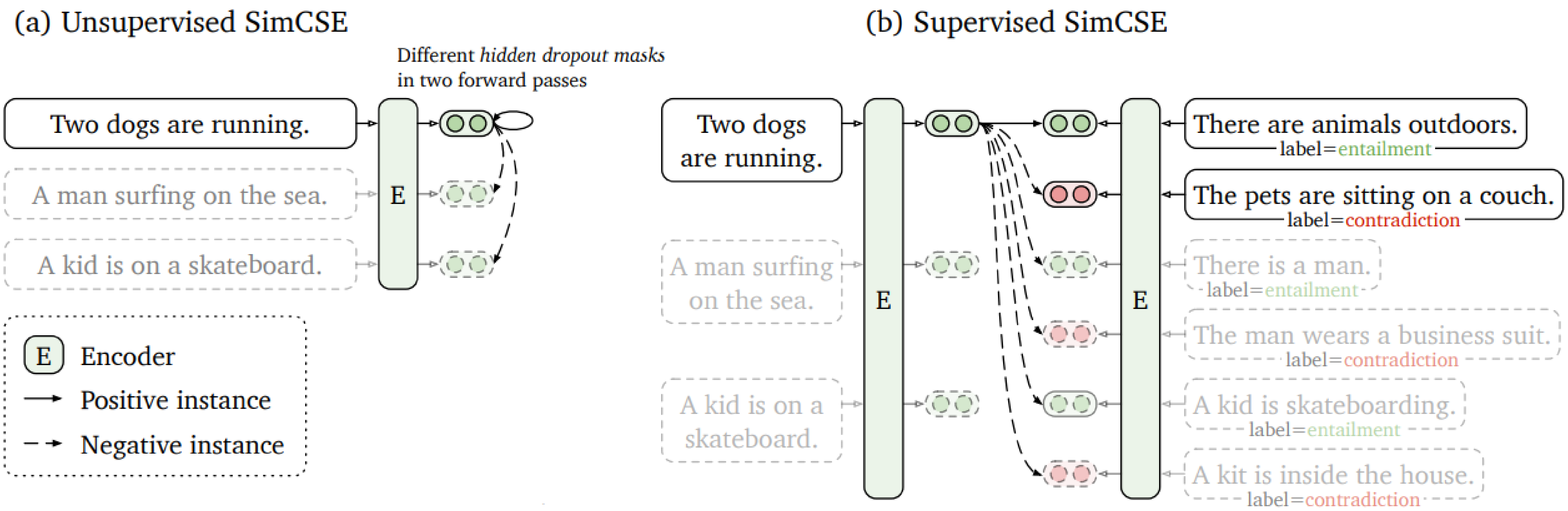


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

Model	STS12	STS13	STS14	STS15	STS16	STSb	SICK-R	Avg.
Avg. GloVe embeddings	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
Avg. BERT embeddings	38.78	57.98	57.98	63.15	61.06	46.35	58.40	54.81
BERT CLS-vector	20.16	30.01	20.09	36.88	38.08	16.50	42.63	29.19
InferSent - Glove	52.86	66.75	62.15	72.77	66.87	68.03	65.65	65.01
Universal Sentence Encoder	64.49	67.80	64.61	76.83	73.18	74.92	**76.69**	71.22
SBERT-NLI-base	70.97	76.53	73.19	79.09	74.30	77.03	72.91	74.89
SBERT-NLI-large	72.27	**78.46**	**74.90**	80.99	76.25	**79.23**	73.75	76.55
SRoBERTa-NLI-base	71.54	72.49	70.80	78.74	73.69	77.77	74.46	74.21
SRoBERTa-NLI-large	**74.53**	77.00	73.18	**81.85**	**76.82**	79.10	74.29	**76.68**

Table 1: Spearman rank correlation ρ between the cosine similarity of sentence representations and the gold labels for various Textual Similarity (STS) tasks. Performance is reported by convention as $\rho \times 100$. STS12-STS16: SemEval 2012-2016, STSb: STSbenchmark, SICK-R: SICK relatedness dataset.

SimCSE: Contrastive Learning

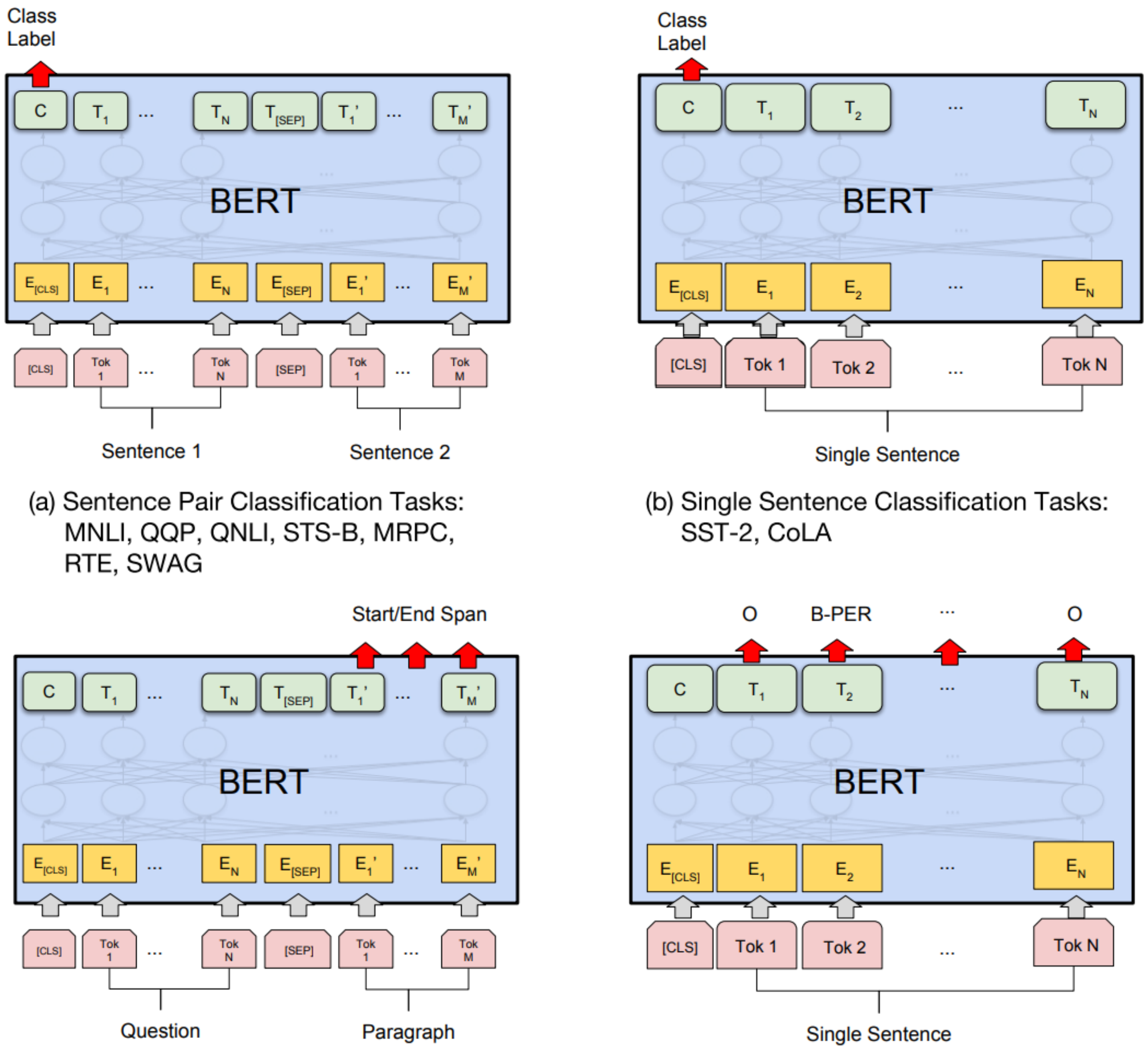


$$\ell_i = -\log \frac{e^{\mathrm{sim}(\mathbf{h}_i, \mathbf{h}_i^+)/\tau}}{\sum_{j=1}^{N} e^{\mathrm{sim}(\mathbf{h}_i, \mathbf{h}_j^+)/\tau}},$$

Model	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
Unsupervised models								
GloVe embeddings (avg.)♣	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
$\text{BERT}_{\texttt{base}}$ (first-last avg.)	39.70	59.38	49.67	66.03	66.19	53.87	62.06	56.70
$\text{BERT}_{\texttt{base}}$-flow	58.40	67.10	60.85	75.16	71.22	68.66	64.47	66.55
$\text{BERT}_{\texttt{base}}$-whitening	57.83	66.90	60.90	75.08	71.31	68.24	63.73	66.28
IS-$\text{BERT}_{\texttt{base}}$♡	56.77	69.24	61.21	75.23	70.16	69.21	64.25	66.58
CT-$\text{BERT}_{\texttt{base}}$	61.63	76.80	68.47	77.50	76.48	74.31	69.19	72.05
∗ SimCSE-$\text{BERT}_{\texttt{base}}$	**68.40**	**82.41**	**74.38**	**80.91**	**78.56**	**76.85**	**72.23**	**76.25**
$\text{RoBERTa}_{\texttt{base}}$ (first-last avg.)	40.88	58.74	49.07	65.63	61.48	58.55	61.63	56.57
$\text{RoBERTa}_{\texttt{base}}$-whitening	46.99	63.24	57.23	71.36	68.99	61.36	62.91	61.73
DeCLUTR-$\text{RoBERTa}_{\texttt{base}}$	52.41	75.19	65.52	77.12	78.63	72.41	**68.62**	69.99
∗ SimCSE-$\text{RoBERTa}_{\texttt{base}}$	**70.16**	**81.77**	**73.24**	**81.36**	**80.65**	**80.22**	68.56	**76.57**
∗ SimCSE-$\text{RoBERTa}_{\texttt{large}}$	**72.86**	**83.99**	**75.62**	**84.77**	**81.80**	**81.98**	**71.26**	**78.90**
Supervised models								
InferSent-GloVe♣	52.86	66.75	62.15	72.77	66.87	68.03	65.65	65.01
Universal Sentence Encoder♣	64.49	67.80	64.61	76.83	73.18	74.92	76.69	71.22
$\text{SBERT}_{\texttt{base}}$♣	70.97	76.53	73.19	79.09	74.30	77.03	72.91	74.89
$\text{SBERT}_{\texttt{base}}$-flow	69.78	77.27	74.35	82.01	77.46	79.12	76.21	76.60
$\text{SBERT}_{\texttt{base}}$-whitening	69.65	77.57	74.66	82.27	78.39	79.52	76.91	77.00
CT-$\text{SBERT}_{\texttt{base}}$	74.84	83.20	78.07	83.84	77.93	81.46	76.42	79.39
∗ SimCSE-$\text{BERT}_{\texttt{base}}$	**75.30**	**84.67**	**80.19**	**85.40**	**80.82**	**84.25**	**80.39**	**81.57**
$\text{SRoBERTa}_{\texttt{base}}$♣	71.54	72.49	70.80	78.74	73.69	77.77	74.46	74.21
$\text{SRoBERTa}_{\texttt{base}}$-whitening	70.46	77.07	74.46	81.64	76.43	79.49	76.65	76.60
∗ SimCSE-$\text{RoBERTa}_{\texttt{base}}$	**76.53**	**85.21**	**80.95**	**86.03**	**82.57**	**85.83**	**80.50**	**82.52**
∗ SimCSE-$\text{RoBERTa}_{\texttt{large}}$	**77.46**	**87.27**	**82.36**	**86.66**	**83.93**	**86.70**	**81.95**	**83.76**

Why is BERT great?

- All NLP task:
  - One of these four cases
  - Or some clever adaptation (Assignment 1 Q1/2)
  - Very good result!

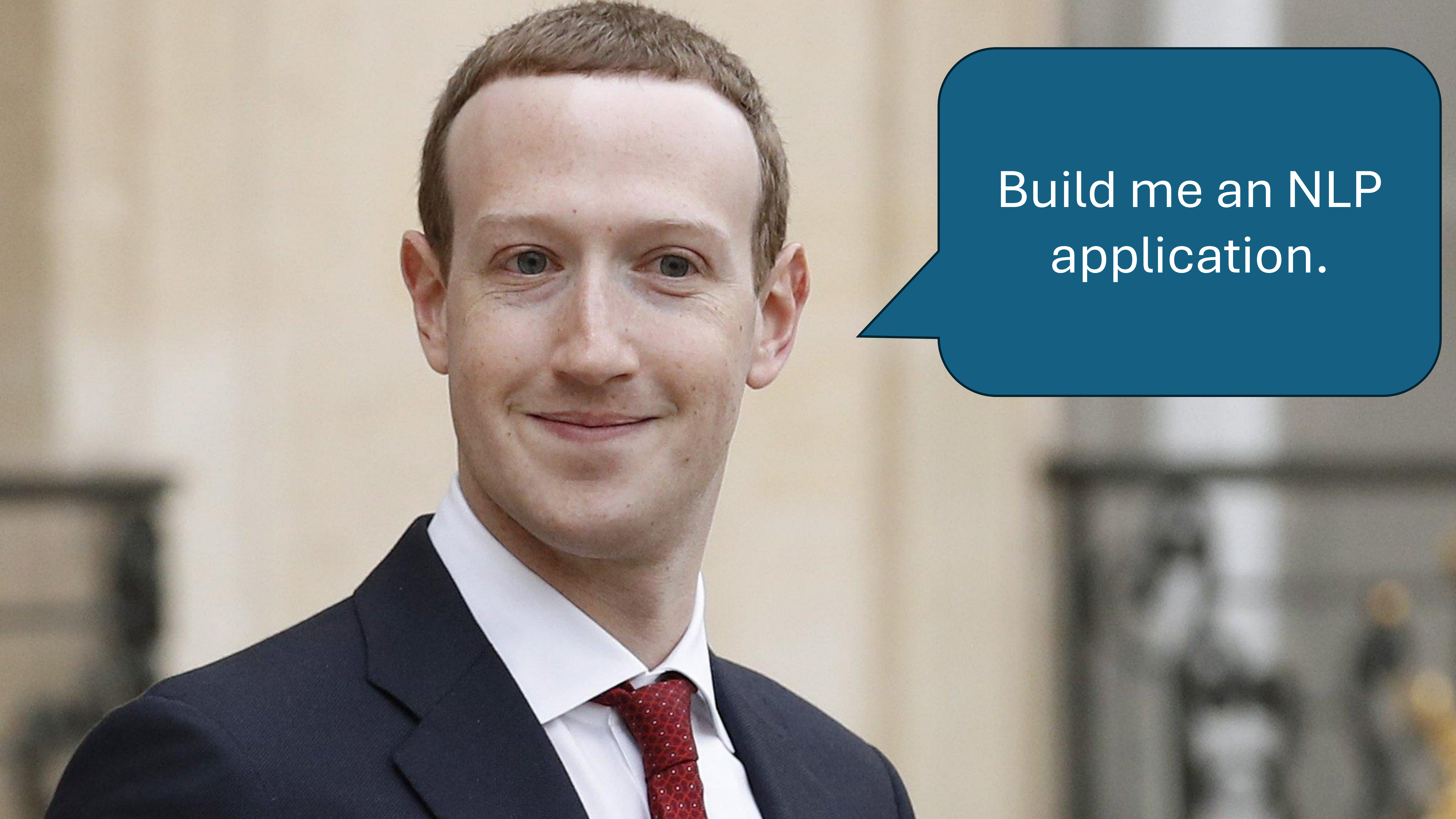


(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

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

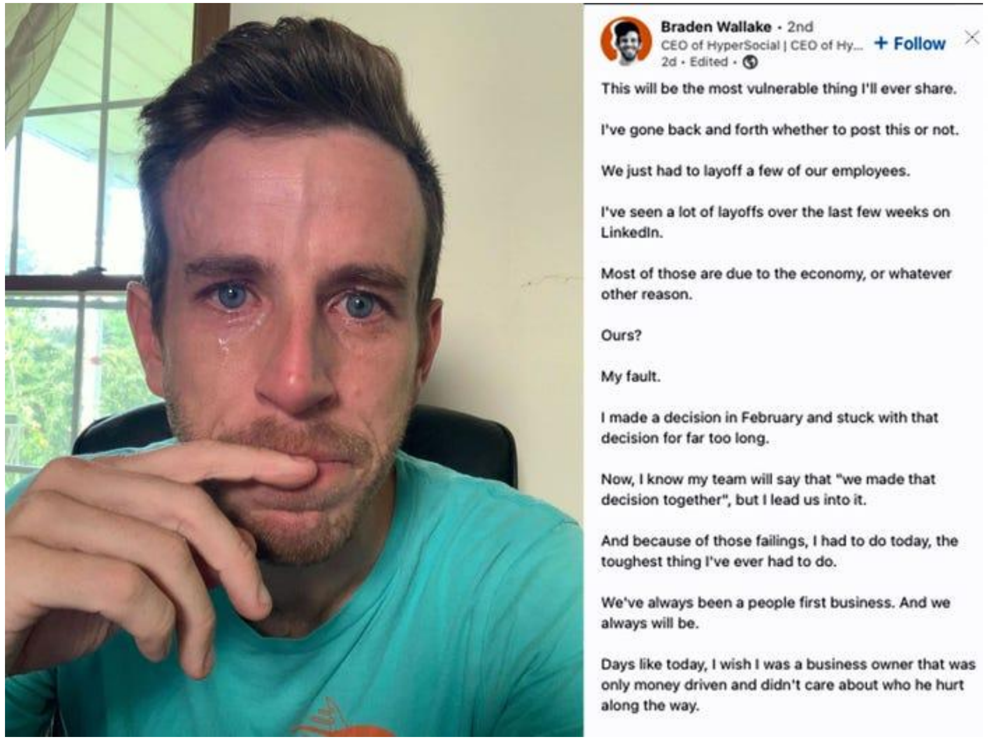
(c) Question Answering Tasks: SQuAD v1.1

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


Build me an NLP application.

```python
import torch
from datasets import load_dataset
from transformers import AutoTokenizer, AutoModelForSequenceClassification, TrainingArguments, Trainer

if __name__ == '__main__':
    dataset = load_dataset("yelp_review_full")
    tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")

    def tokenize_function(examples):
        return tokenizer(examples["text"], padding="max_length", truncation=True)

    tokenized_datasets = dataset.map(tokenize_function, batched=True)
    model = AutoModelForSequenceClassification.from_pretrained("bert-base-cased", num_labels=5)

    def acc(eval_pred):
        logits, labels = eval_pred
        predictions = torch.argmax(logits, dim=-1)
        return (predictions == labels).sum().item() / len(labels)

    training_args = TrainingArguments(output_dir="test_trainer", report_to=None)

    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=tokenized_datasets["train"],
        eval_dataset=tokenized_datasets["test"],
        compute_metrics=acc,
    )

    trainer.train()
```

NOW, ENJOY YOUR


Braden Wallake · 2nd
CEO of HyperSocial | CEO of Hy...
2d · Edited ·
+ Follow

This will be the most vulnerable thing I'll ever share.

I've gone back and forth whether to post this or not.

We just had to layoff a few of our employees.

I've seen a lot of layoffs over the last few weeks on LinkedIn.

Most of those are due to the economy, or whatever other reason.

Ours?

My fault.

I made a decision in February and stuck with that decision for far too long.

Now, I know my team will say that "we made that decision together", but I lead us into it.

And because of those failings, I had to do today, the toughest thing I've ever had to do.

We've always been a people first business. And we always will be.

Days like today, I wish I was a business owner that was only money driven and didn't care about who he hurt along the way.