Deep Learning for Vision & Language

Natural Language Processing II: Representations/Tokenization

RICE UNIVERSITY

How to represent a word?

		one-hot encodings
dog	1	[1 0 0 0 0 0 0 0 0 0]
cat	2	[0 1 0 0 0 0 0 0 0 0]
person	3	[0 0 1 0 0 0 0 0 0 0]
holding	4	[0 0 0 1 0 0 0 0 0 0]
tree	5	[0 0 0 0 1 0 0 0 0 0]
computer	6	[0 0 0 0 0 1 0 0 0 0]
using	7	[0 0 0 0 0 0 1 0 0 0]

How to represent a word?

How to represent a phrase/sentence?

bag-of-words representation

person holding dog	{1, 3, 4}	[1	0	1	1	0	0	0	0	0	0]
person holding cat	{2, 3, 4}	[0	1	1	1	0	0	0	0	0	0]
person using computer	{3, 7, 6}	[0	0	1	0	0	1	1	0	0	0]
		dog	cat	person	holding	tree	computer	using			

person using computer person holding cat	{3, 3, 7, 6, 2}	[0	1	2	1	0	1	1	0	0	0]

What if vocabulary is very large?

Sparse Representation

bag-of-words representation

person holding dog	{1, 3, 4}	indices = [1, 3, 4]	values = [1, 1, 1]
person holding cat	{2, 3, 4}	indices = [2, 3, 4]	values = [1, 1, 1]
person using computer	{3, 7, 6}	indices = [3, 7, 6]	values = [1, 1, 1]

person using computer
person holding cat

{3, 3, 7, 6, 2}

indices = [3, 7, 6, 2] values = [2, 1, 1, 1]

Recap

- Bag-of-words encodings for text (e.g. sentences, paragraphs, captions, etc)

You can take a set of sentences/documents and classify them, cluster them, or compute distances between them using this representation.

Problem with this bag-of-words representation

my friend makes a nice meal

These would be the same using bag-of-words

my nice friend makes a meal

Bag of Bi-grams

my friend makes a nice meal

indices = [10132, 21342, 43233, 53123, 64233]
values = [1, 1, 1, 1, 1]

{my friend, friend makes, makes a, a nice, nice meal}

my nice friend makes a meal

indices = [10232, 43133, 21342, 43233, 54233]
values = [1, 1, 1, 1, 1]

{my nice, nice friend, friend makes, makes a, a meal}

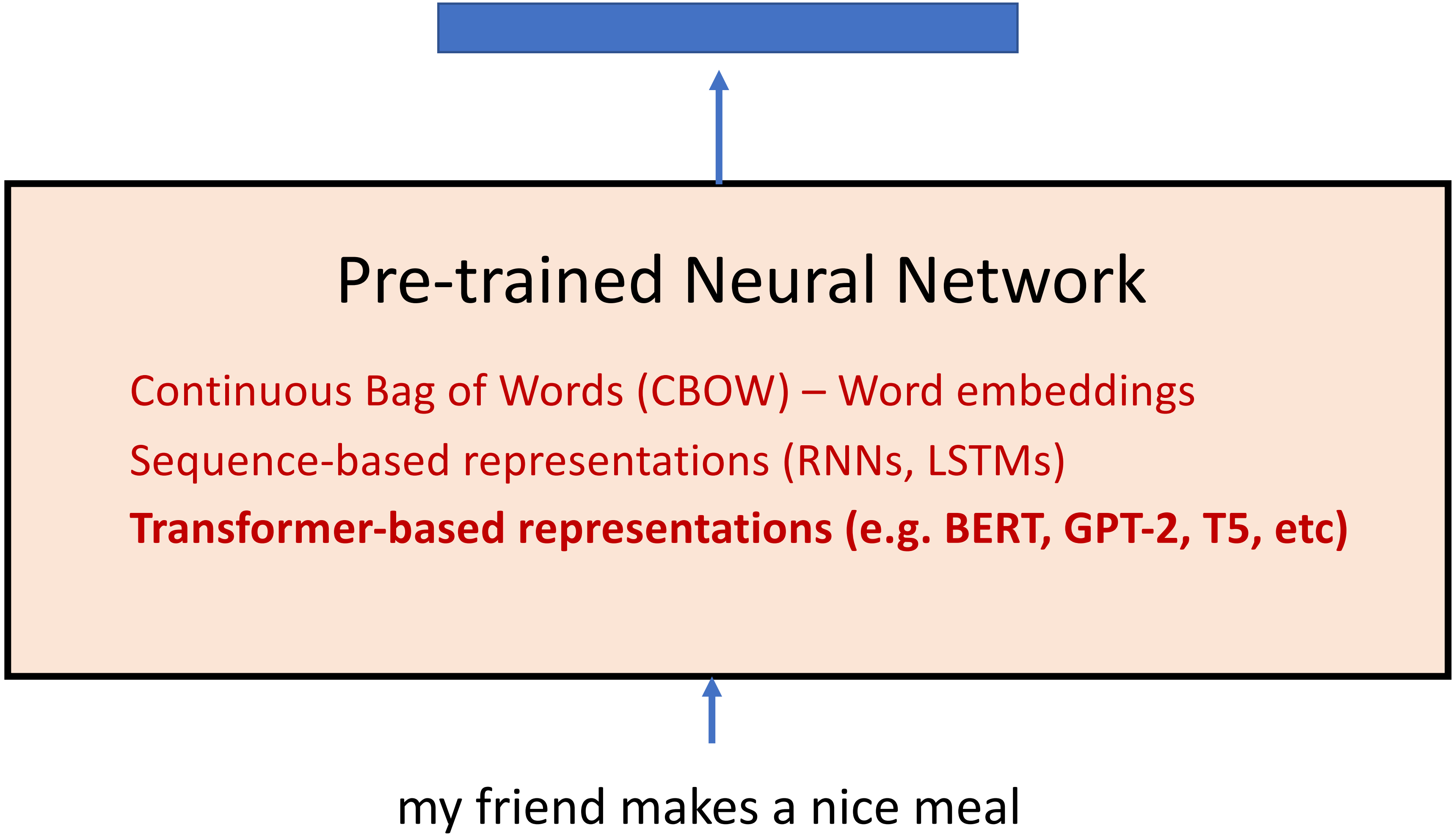
A dense vector-representation would be very inefficient
Think about tri-grams and n-grams

Recommended reading: n-gram language models

Yejin Choi's course on Natural Language Processing

http://www3.cs.stonybrook.edu/~ychoi/cse628/lecture/02-ngram.pdf

Modern way of representing Phrases/Text



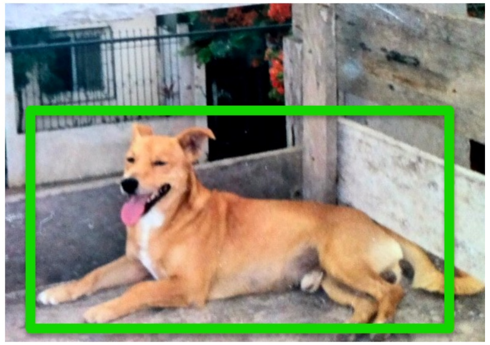
Back to how to represent a word?

Problem: distance between words using one-hot encodings always the same

dog	1	[1 0 0 0 0 0 0 0 0 0]
cat	2	[0 1 0 0 0 0 0 0 0 0]
person	3	[0 0 1 0 0 0 0 0 0 0]

Idea: Instead of one-hot-encoding use a histogram of commonly co-occurring words.

Distributional Semantics



Dogs are man’s best friend.

I saw a dog on a leash walking in the park.

His dog is his best companion.

He walks his dog in the late afternoon

…

	friend	leash	park	walking	walks	food	legs	runs	sleeps	sits	…
dog	[3	2	3	4	2	4	3	5	6	7	…]

Distributional Semantics

	food	walks	window	runs	mouse	invented	legs	sleeps	mirror	tail	...
dog	[5	5	0	5	0	0	5	5	0	2	...]
cat	[5	4	1	4	2	0	3	4	0	3	...]
person	[5	5	1	5	0	2	5	5	0	0	...]

This vocabulary can be extremely large

Toward more Compact Representations

dog	[5	5	0	5	0	0	5	5	0	2	...]
cat	[5	$\boxed{4}$	1	$\boxed{4}$	2	0	3	$\boxed{4}$	0	3	...]
person	[5	5	1	5	0	2	5	5	0	0	...]
	food	walks	window	runs	mouse	invented	legs	sleeps	mirror	tail	...

This vocabulary can be extremely large

Toward more Compact Representations

$$\text{dog} = \begin{bmatrix} 5 \\ 5 \\ 0 \\ 5 \\ 0 \\ 0 \\ 5 \\ 5 \\ 0 \\ 2 \\ \dots \end{bmatrix} = w1 \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \dots \end{bmatrix} + w2 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ \dots \end{bmatrix} + w3 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ \dots \end{bmatrix} + \dots$$

legs, running, walking

tail, fur, ears

mirror, window, door

Toward more Compact Representations

dog = $\begin{bmatrix} w1 & w2 & w3 \end{bmatrix}$

The basis vectors can be found using Principal Component Analysis (PCA)

This is known as Latent Semantic Analysis in NLP

Toward more Compact Representations: Word Embeddings

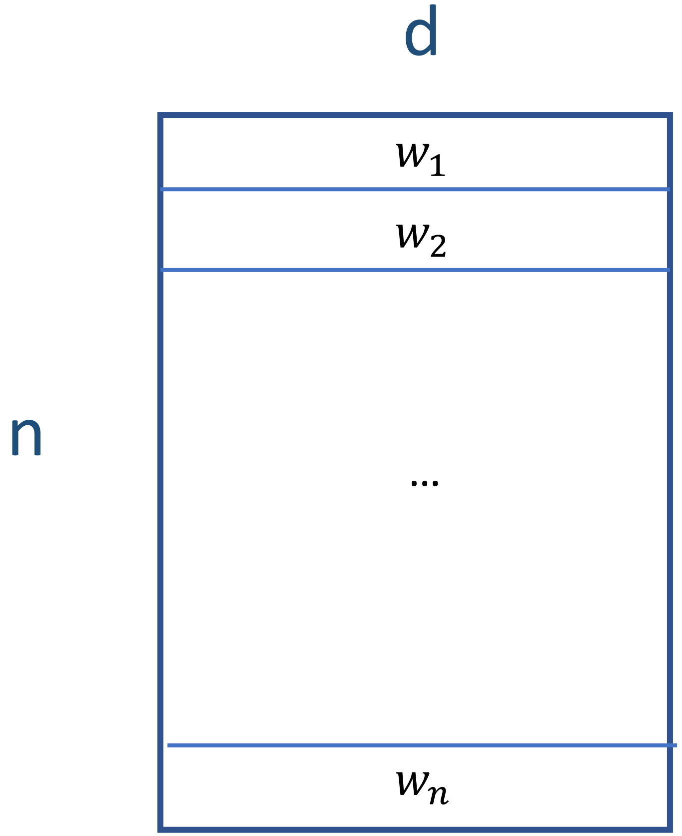
dog = [w1 w2 w3]

The weights w1, …, wn are found using a neural network

Word2Vec: https://arxiv.org/abs/1301.3781

Word2Vec – CBOW Version

- First, create a huge matrix of word embeddings initialized with random values – where each row is a vector for a different word in the vocabulary.



Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

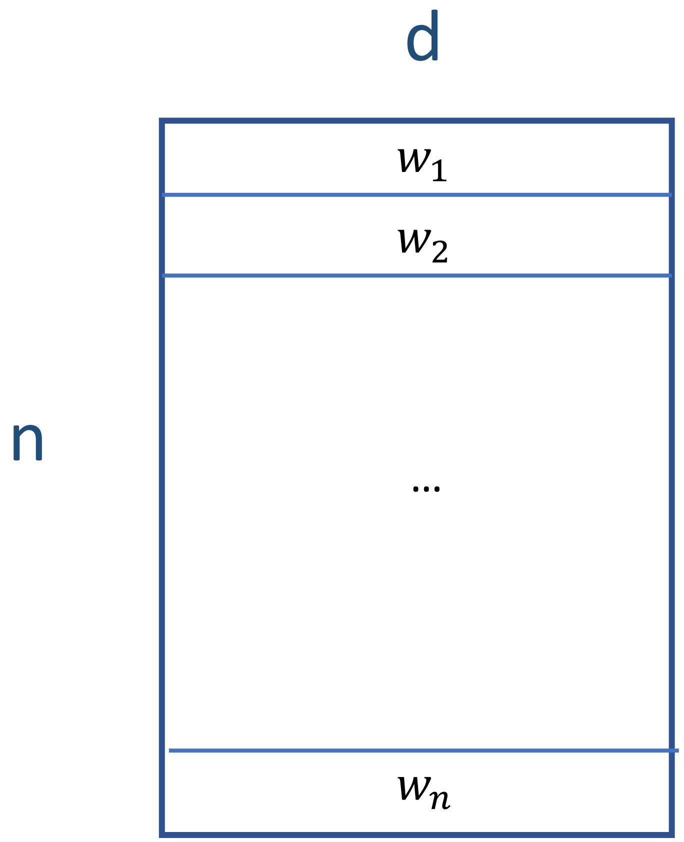
Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

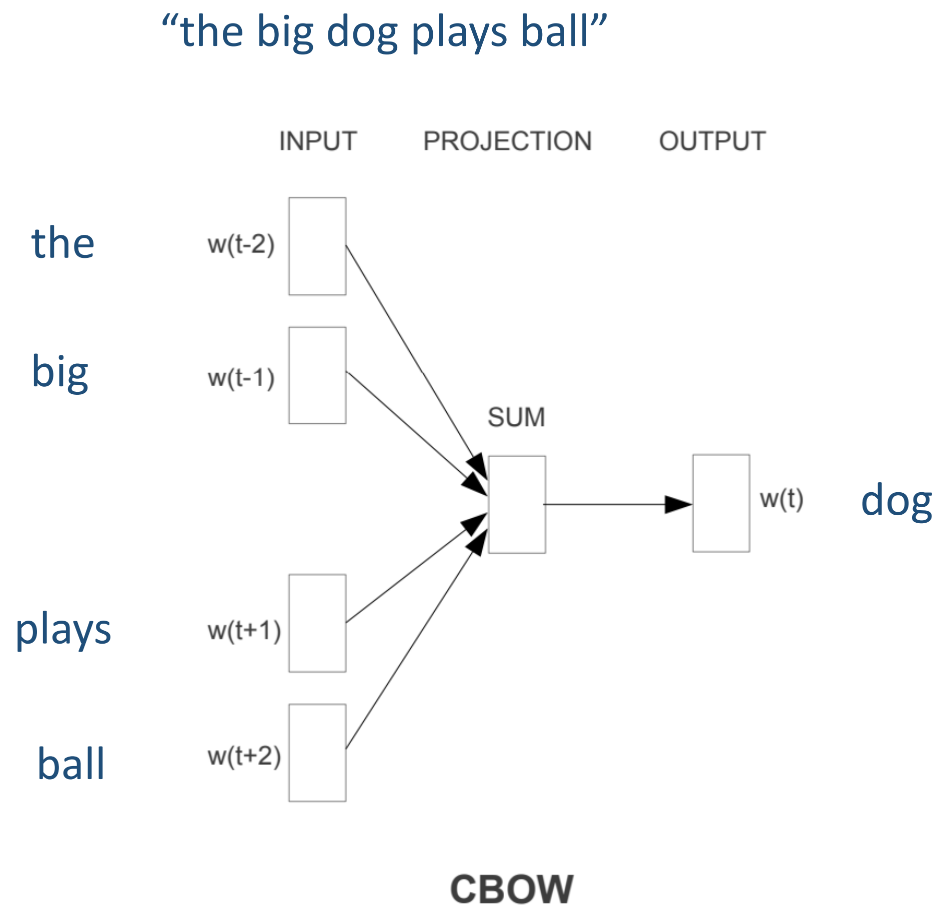
Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

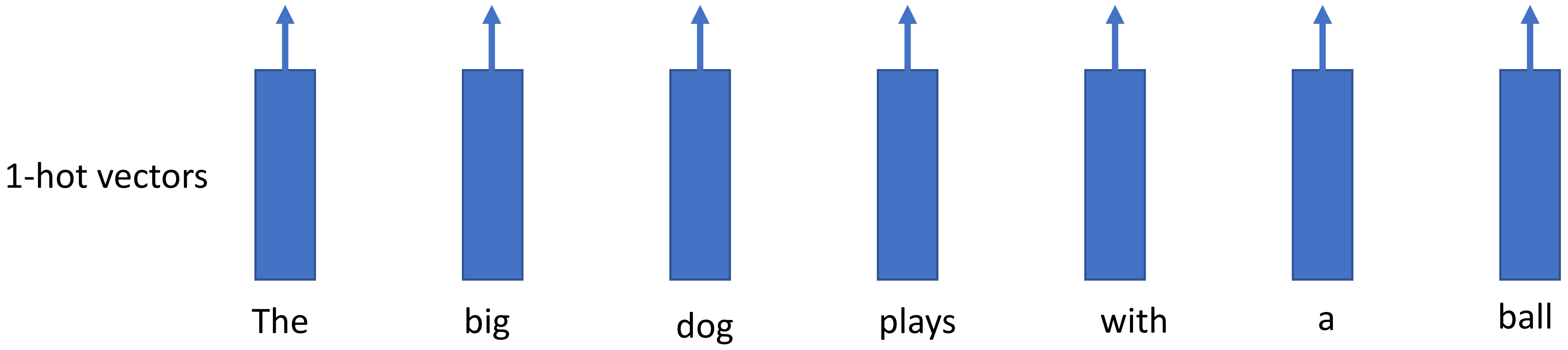
Word2Vec – CBOW Version

- Then, collect a lot of text, and solve the following regression problem for a large corpus of text:

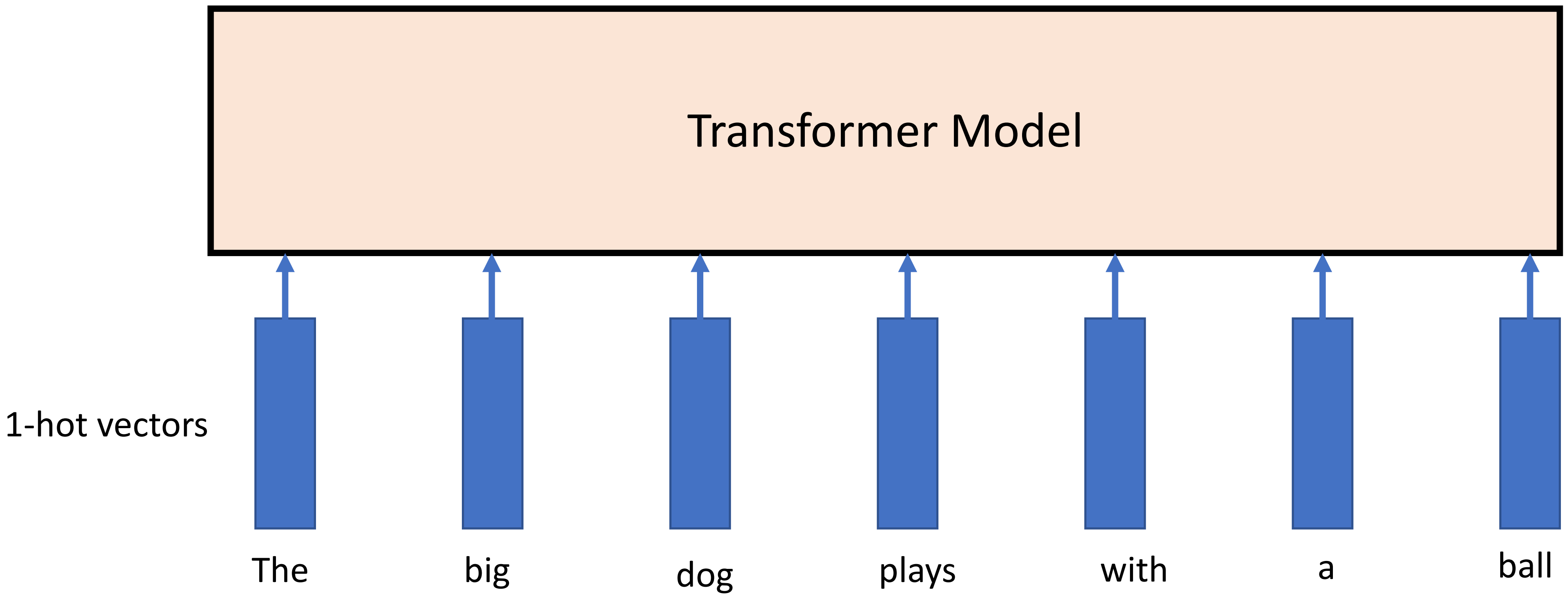




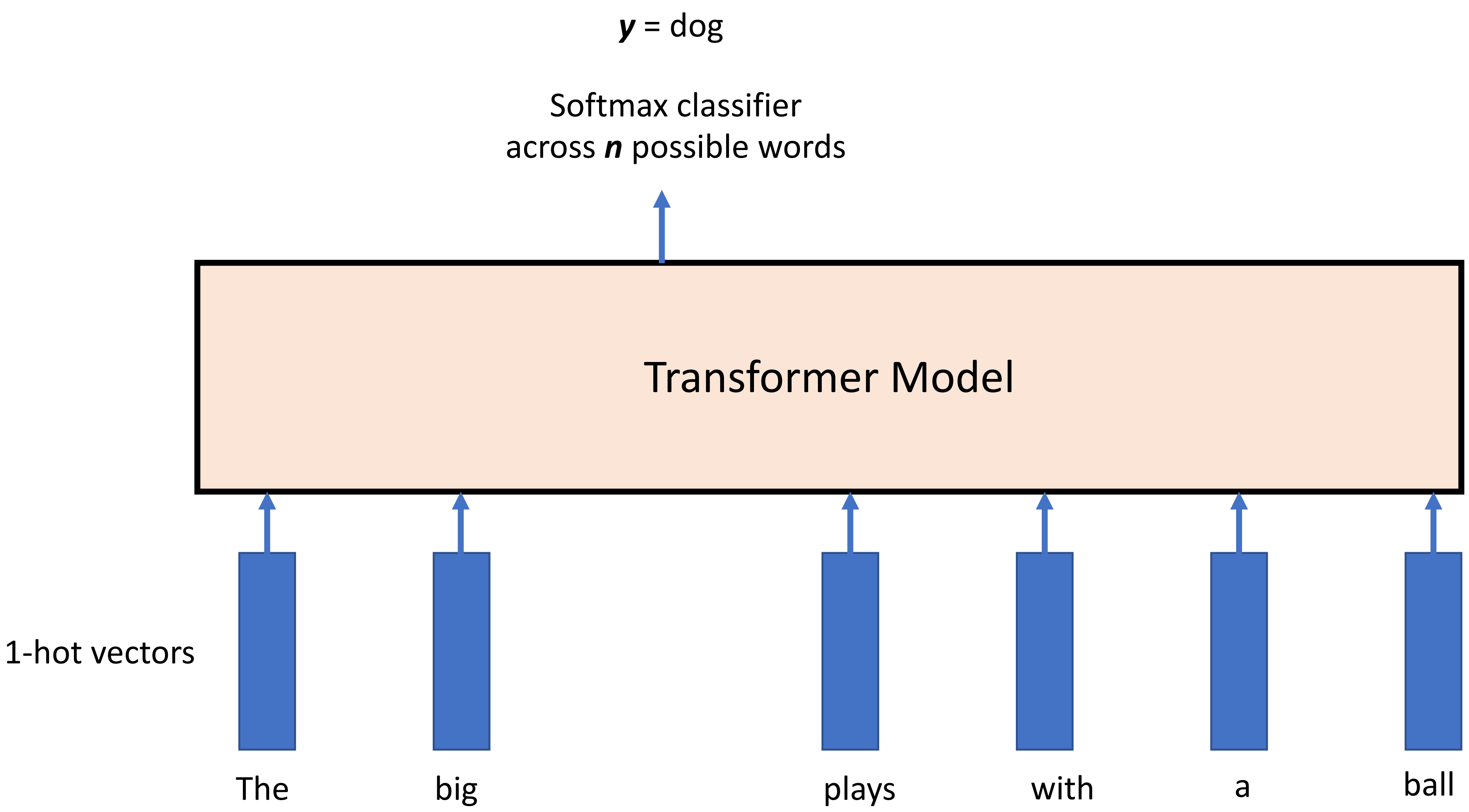
Pre-trained Language Models



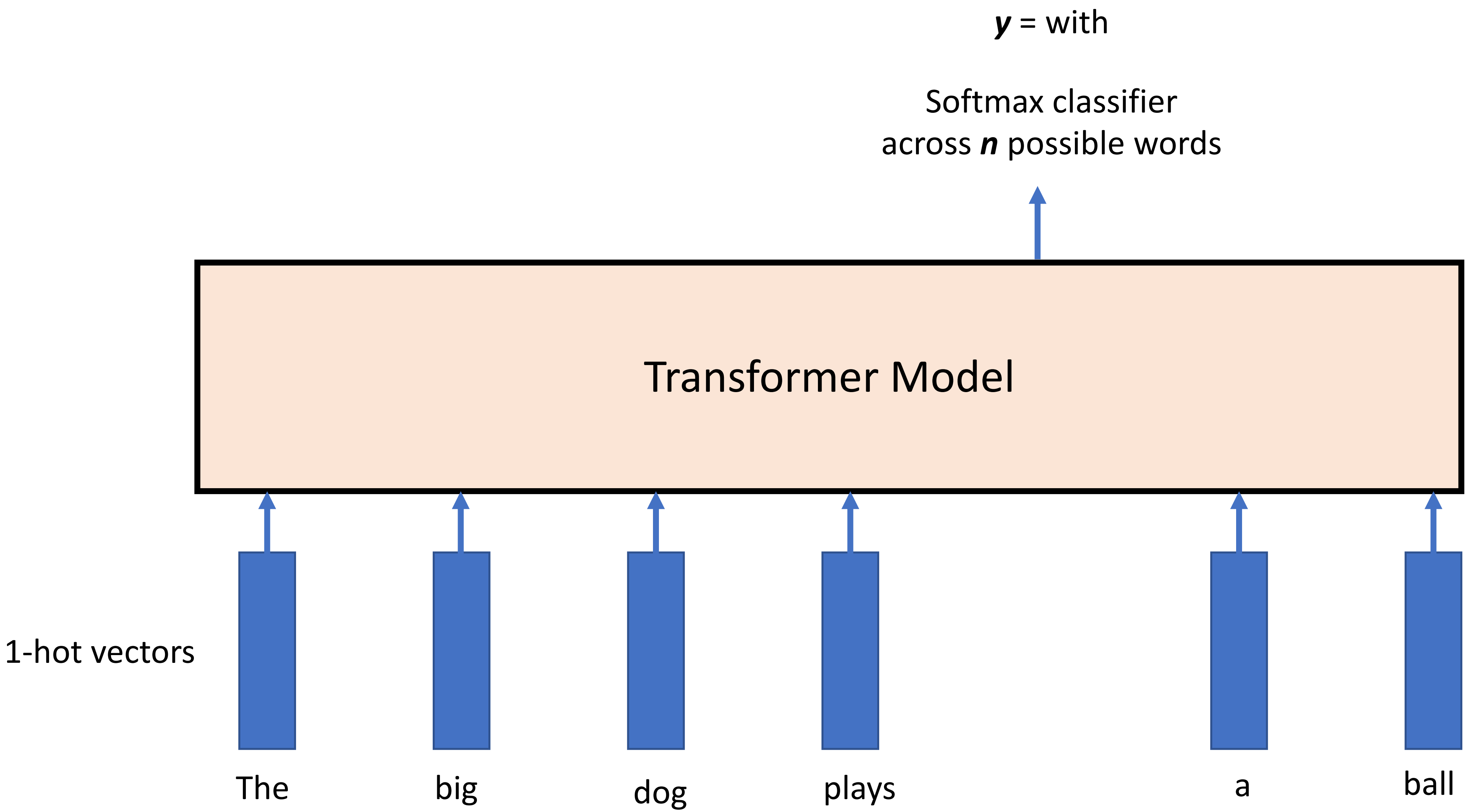
Pre-trained Language Models



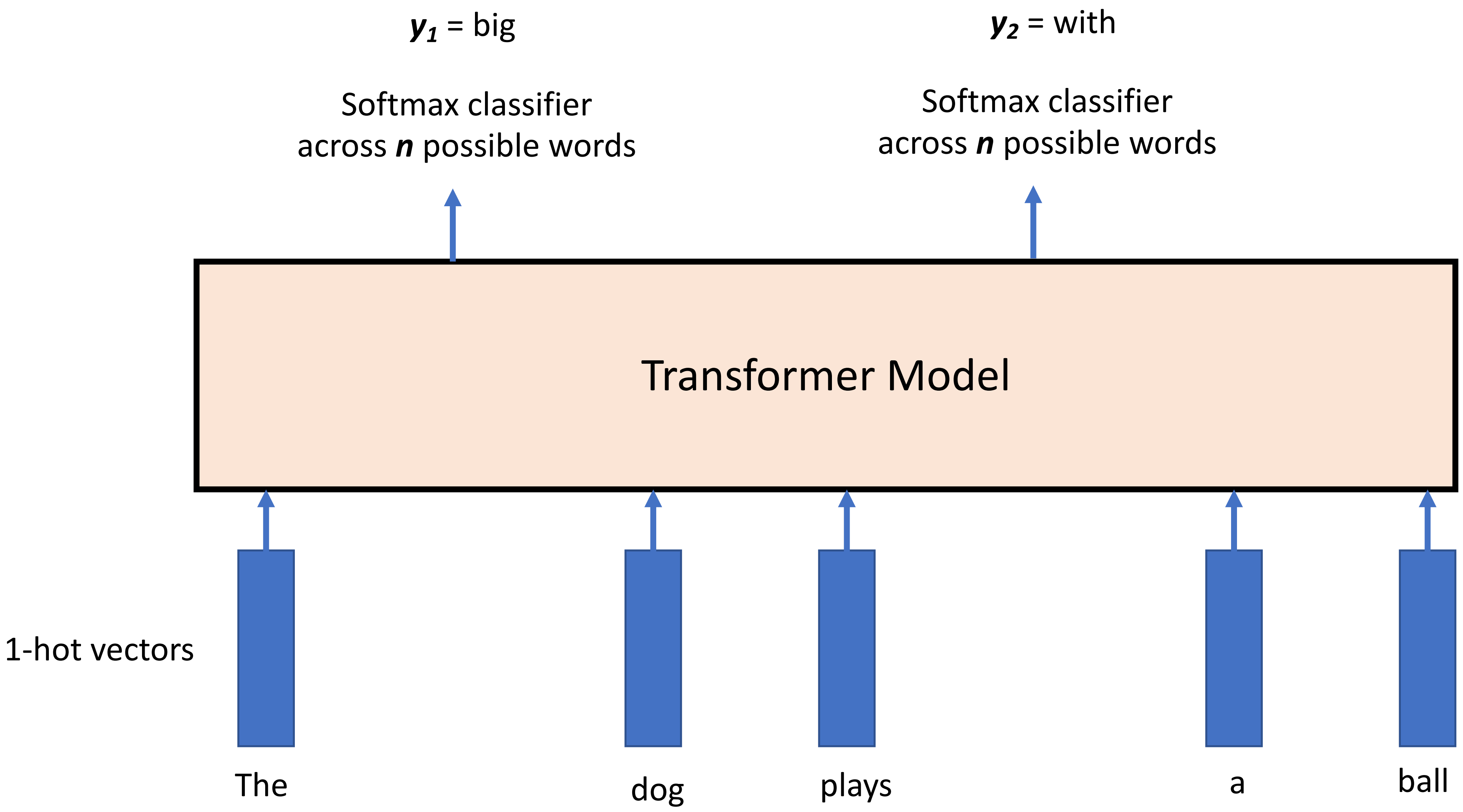
Pre-trained Language Models



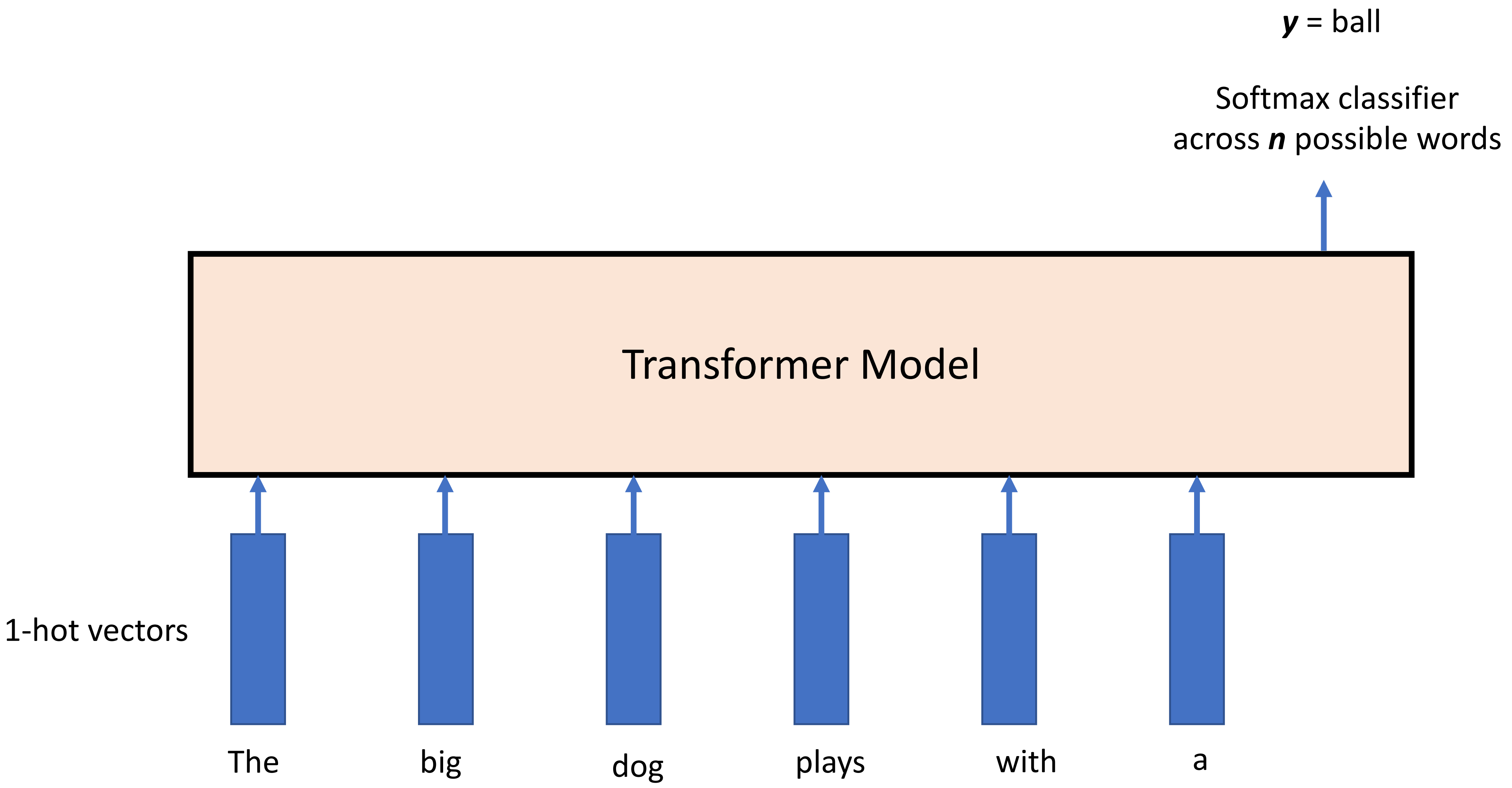
Pre-trained Language Models



Pre-trained Language Models



Generative Language Models



Practical Issues - Tokenization

- For each text representation we usually need to separate a sentence into tokens – we have assumed words in this lecture (or pairs of words) – but tokens could also be characters and anything in-between.

- Word segmentation can be used as tokenization.
  - In the assignment I was lazy I just did “my sentence”.split(“ “) and called it a day.
  - However, even English is more difficult than that because of punctuation, double spaces, quotes, etc. For English I would recommend you too look up the great word tokenization tools in libraries such as Python’s NLTK and Spacy before you try to come up with your own word tokenizer.

Issues with Word based Tokenization

- We already mentioned that tokenization can be hard even when word-based for other languages that don't use spaces in-between words.

- Word tokenization can also be bad for languages where the words can be "glued" together like German or Turkish.
  - Remember fünfhundertfünfundfünfzig? It wouldn't be feasible to have a word embedding for every number in the German language.

- It is problematic to handle words that are not in the vocabulary e.g. a common practice is to use a special <OOV> (out of vocabulary) token for those words that don't show up in the vocabulary.

Tokenization can be complex

- Think of Japanese
  - Three vocabularies/sets of symbols:
    Katakana and Hiragana symbols represent syllables / sounds
    く = ku, ぎ = gi, ナ = na, ア= a
    Kanji represent ideas / words (Chinese characters).
    日 = day, sun, 大 = big, 凸= convex 凹 = concave
  - They can be combined – e.g. tomorrow = 明日
  - Each symbol also has some structure within the symbols. They are not independently created. e.g. bright= 明るい , rising sun = 旭
  - And of course there are no spaces in between the characters.

Solution: Sub-word Tokenization

- Byte-pair Encoding Tokenization (BPE)
  - Start from small strings and based on substring counts iteratively use larger sequences until you define a vocabulary that maximizes informative subtokens. That way most will correspond to words at the end.
- Byte-level BPE Tokenizer
  - Do the same but at the byte representation level not at the substring representation level.

We will discuss these more as we discuss Transformer Models

Tokenizers

Rust passing | license Apache-2.0 | downloads/week 169k

Provides an implementation of today's most used tokenizers, with a focus on performance and versatility.

Main features:

- Train new vocabularies and tokenize, using today's most used tokenizers.
- Extremely fast (both training and tokenization), thanks to the Rust implementation. Takes less than 20 seconds to tokenize a GB of text on a server's CPU.
- Easy to use, but also extremely versatile.
- Designed for research and production.
- Normalization comes with alignments tracking. It's always possible to get the part of the original sentence that corresponds to a given token.
- Does all the pre-processing: Truncate, Pad, add the special tokens your model needs.

huggingface/tokenizers

BPE Tokenization Overview

Neural Machine Translation of Rare Words with Subword Units

Rico Sennrich and **Barry Haddow** and **Alexandra Birch**
School of Informatics, University of Edinburgh
{rico.sennrich,a.birch}@ed.ac.uk,bhaddow@inf.ed.ac.uk

- Learn BPE operations (python code on the right) – from the paper.
- Use said operations to construct your sub-word vocabulary.
- Treat each sub-word token as a "word" in any models we will discuss.

Algorithm 1 Learn BPE operations

```
import re, collections

def get_stats(vocab):
  pairs = collections.defaultdict(int)
  for word, freq in vocab.items():
    symbols = word.split()
    for i in range(len(symbols)-1):
      pairs[symbols[i],symbols[i+1]] += freq
  return pairs

def merge_vocab(pair, v_in):
  v_out = {}
  bigram = re.escape(' '.join(pair))
  p = re.compile(r'(?<!\S)' + bigram + r'(?!\S)')
  for word in v_in:
    w_out = p.sub(''.join(pair), word)
    v_out[w_out] = v_in[word]
  return v_out

vocab = {'l o w </w>' : 5, 'l o w e r </w>' : 2,
         'n e w e s t </w>':6, 'w i d e s t </w>':3}
num_merges = 10
for i in range(num_merges):
  pairs = get_stats(vocab)
  best = max(pairs, key=pairs.get)
  vocab = merge_vocab(best, vocab)
  print(best)
```

https://colab.research.google.com/drive/1gUjL_h2tXdTtPSfxbBP-6MkE_BMck6gm?usp=sharing

Tokenization used in GPT-3

https://platform.openai.com/tokenizer

The cat is in the house

Tokens 6 Characters 23

The cat is in the house

[464, 3797, 318, 287, 262, 2156]

The geologist made an effort to rationalize the explanation

Tokens 11 Characters 59

The geologist made an effort to rationalize the explanation

[464, 4903, 7451, 925, 281, 3626, 284, 9377, 1096, 262, 7468]

fünfhundertfünfundfünfzig

Tokens 21 Characters 29

fünfhundertfünfundfünfzig

[69, 9116, 77, 69, 3907, 71, 4625, 83, 3907, 69, 9116, 77, 69, 3907, 917, 3907, 69, 9116, 77, 69, 38262]

La ardilla va a la universidad

Tokens 8 Characters 30

La ardilla va a la universidad

[14772, 33848, 5049, 46935, 257, 8591, 5820, 32482]

Tokenization used in GPT-3

https://platform.openai.com/tokenizer

深層学

Tokens 8 | Characters 3

��������

[162, 115, 109, 161, 109, 97, 27764, 99]

কেমন আছেন?

Tokens 20 | Characters 10

��������� ���������?

[48071, 243, 156, 100, 229, 48071, 106, 48071, 101, 220, 48071, 228, 48071, 249, 156, 100, 229, 48071, 101, 30]

வணக்கம்

Tokens 21 | Characters 7

�����������������????

[156, 106, 113, 156, 106, 96, 156, 106, 243, 156, 107, 235, 156, 106, 243, 156, 106, 106, 156, 107, 235]

Questions?