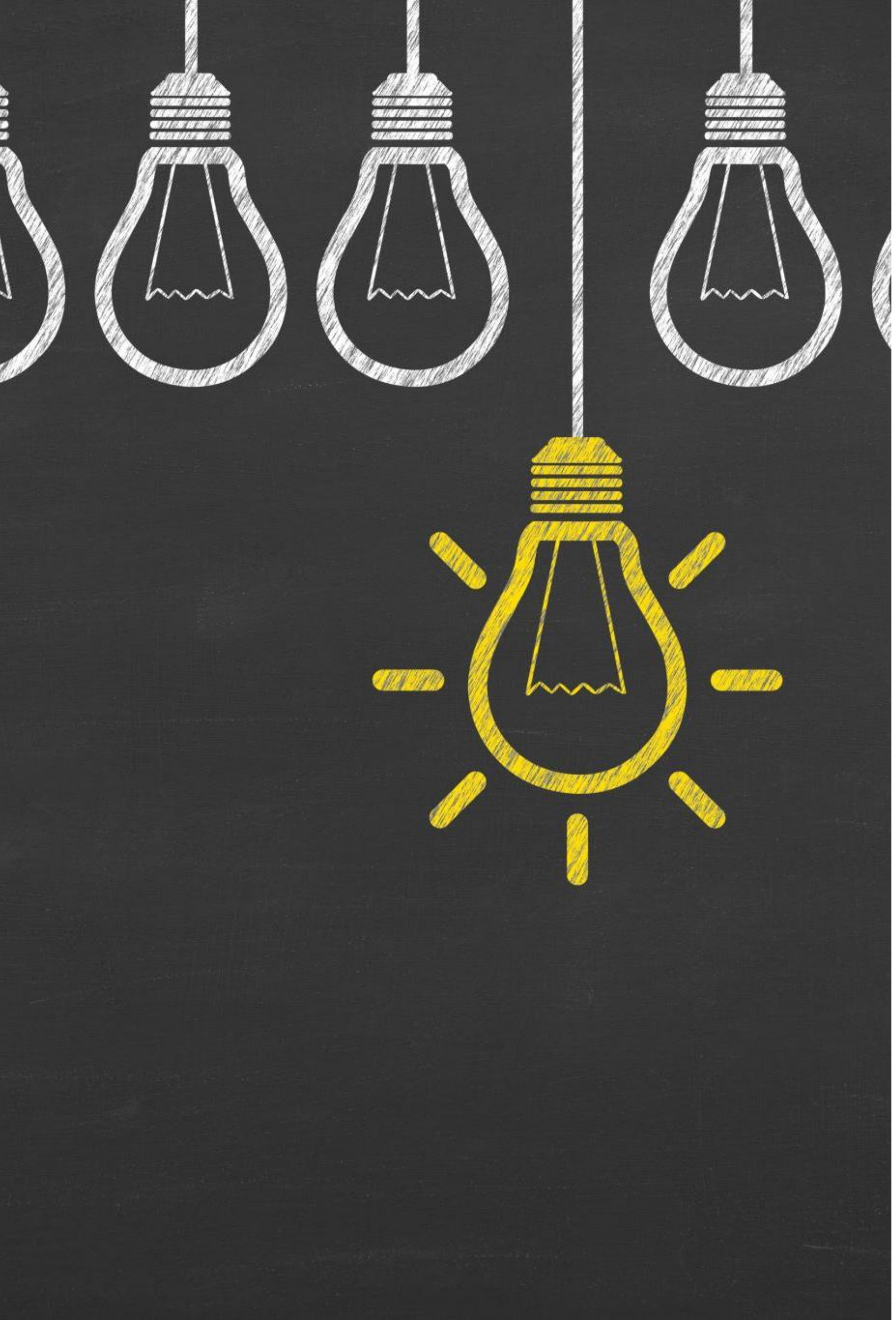
Language Models

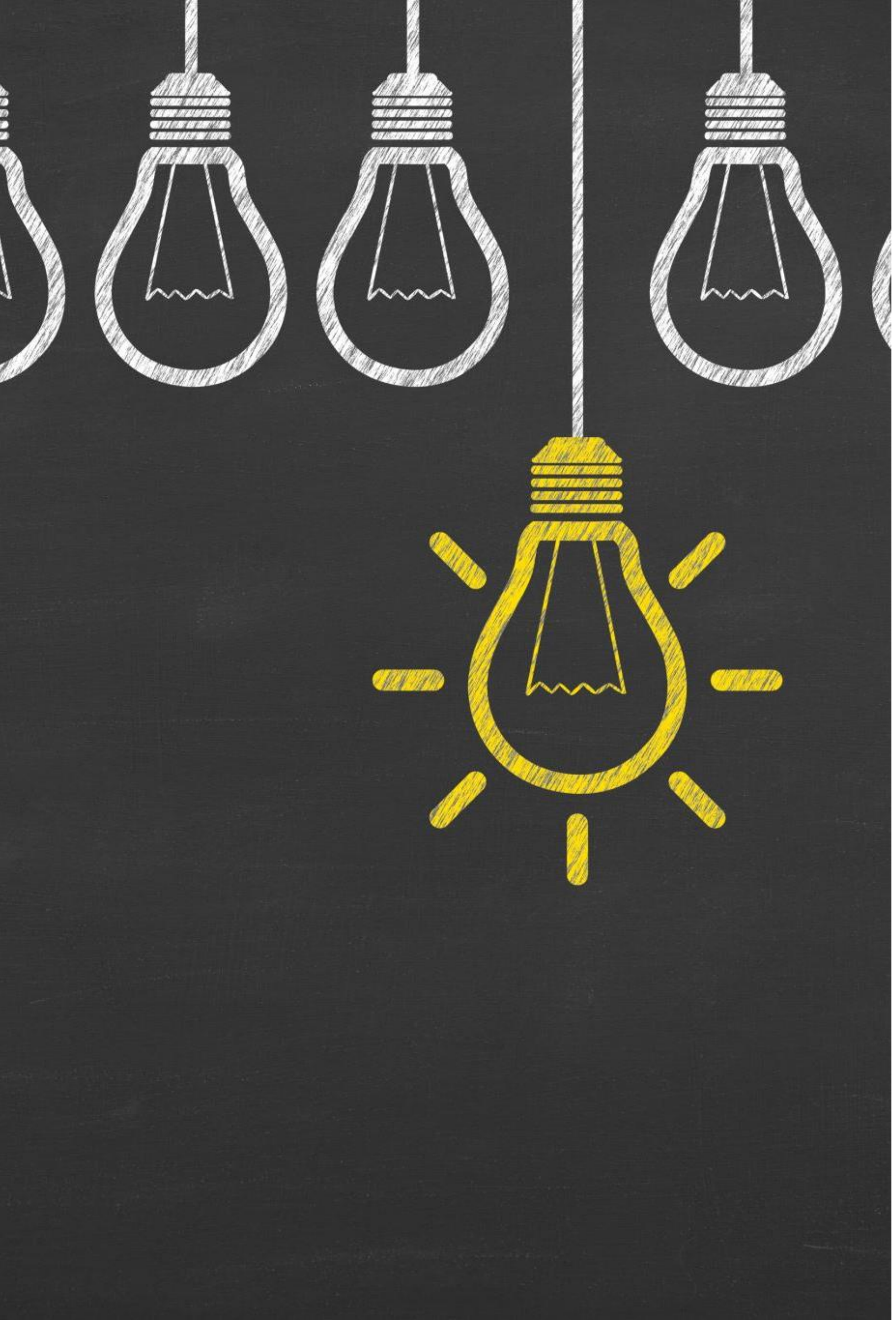
Liad Magen



Sequences

Let's play a game:
what is the next word?

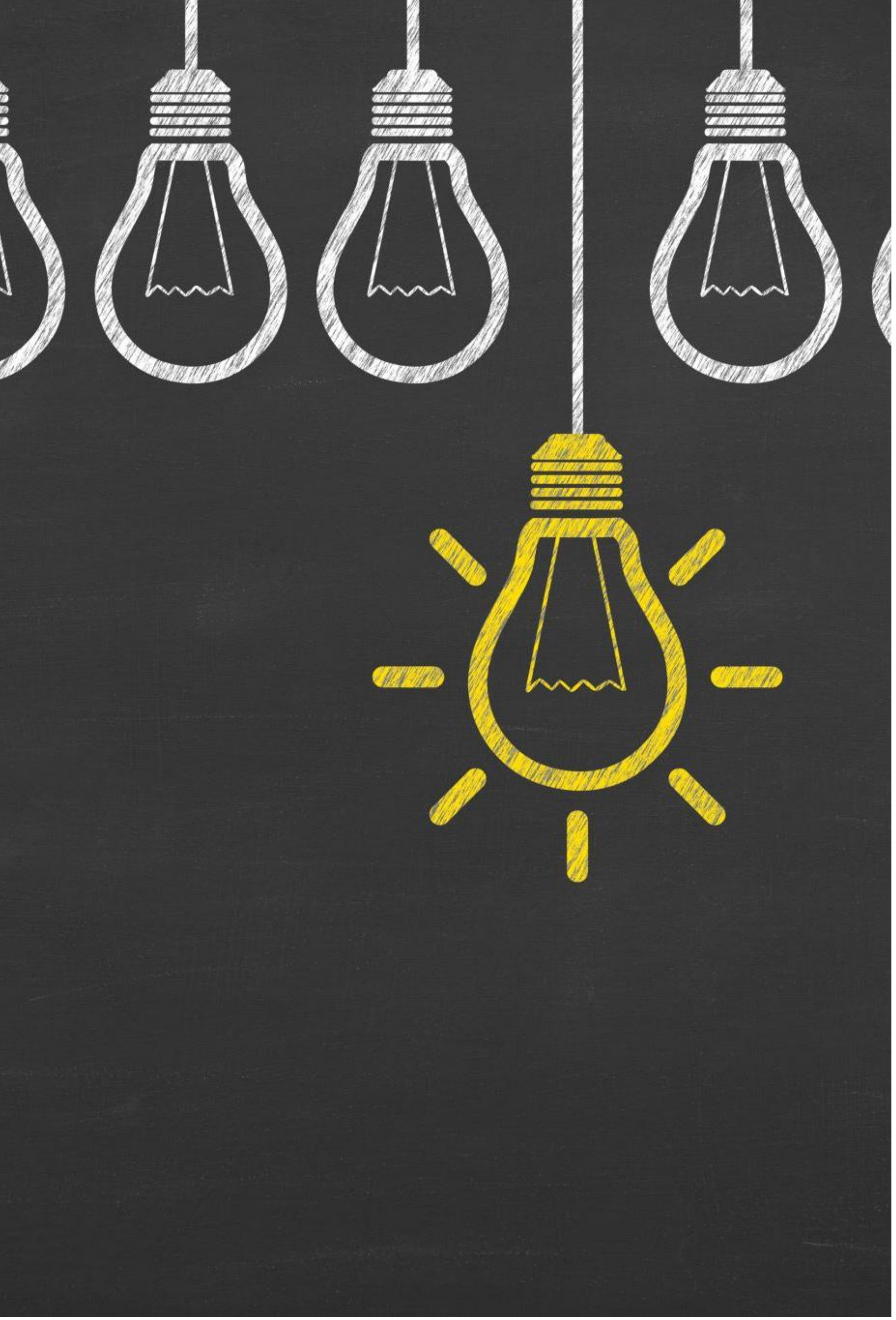
I'm gonna make him an offer he can't ...



Sequences

Let's play a game:
what is the next word?

May the force be with ...



Sequences

Let's play a game:
what is the next word?

There's no place like ...

Language Modeling

Giving a sequence of words (x_1 …. x_{i-1}), compute the probability distribution of the next word.

$$p(x_i | x_1, \ldots, x_{i-1})$$

P(Home | There's no place like)

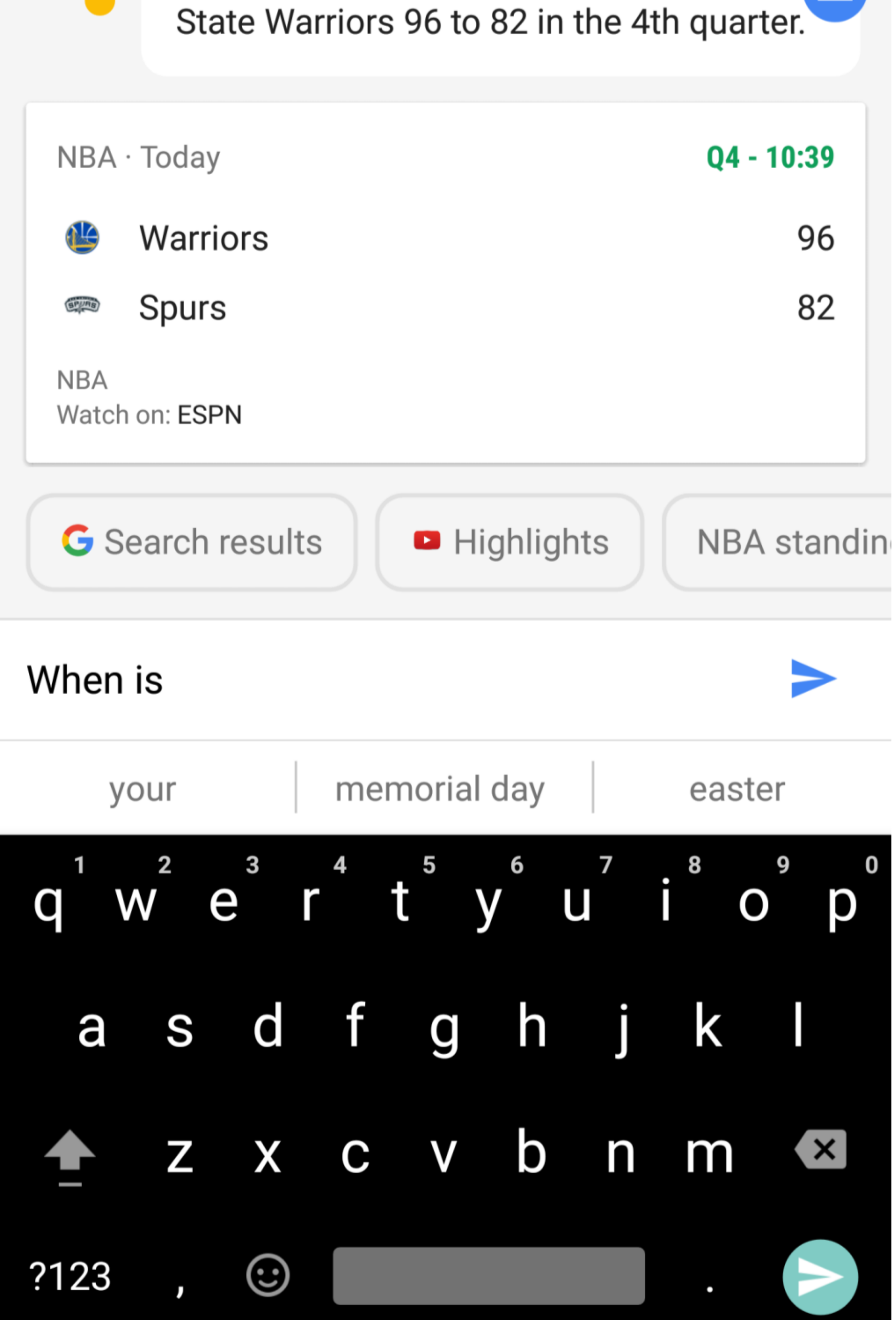
Language Model

Can also assign a *probability score* for a sequence:

$$
\begin{aligned}
p(x_1, ..., x_n) = & p_{LM}(x_1 | *S*, *S*) \\
& \times p_{LM}(x_2 | *S*, x_1) \\
& \times p_{LM}(x_3 | x_1, x_2) \\
& \times p_{LM}(x_4 | x_2, x_3) \\
& \quad \cdots \\
& \times p_{LM}(x_n | x_{n-2}, x_{n-1})
\end{aligned}
$$

P(Their are two examples)

P(There are two examples)



Language Model

- Very useful: also used in *Speech Recognition*, *Machine Translation*, etc.

- ***Note***: Doesn't have to be over natural language. Ideas for other usage examples?

How is it calculated?

- Markov Assumption:
  X_i depends only on the preceding *n-1* words

- n-gram Language Models:

$$p(x_i|x_1, ..., x_{i-1}) = p(x_i / x_{i-n-1}, \ldots , x_{i-1})$$

n-gram Language Models: Example

Suppose n=4:

~~When we collaborate with each other, we~~ can achieve great ________

$$P(\, \mathbf{w} \mid \text{can achieve great}\,) = \frac{\text{count}(\text{can achieve great } \mathbf{w})}{\text{count}(\text{can achieve great})}$$

n-gram Language Models: Sparsity issues

Suppose n=4:

~~When we collaborate with each other, we~~ can achieve great ________

$$P(\mathbf{w} \mid \text{can achieve great}) = \frac{\text{count}(\text{can achieve great } \mathbf{w})}{\text{count}(\text{can achieve great})}$$

What if it never appears and P(W) = 0 ?

What if this n-grams never occurs and the dominator is 0?

Note: The bigger n is, the worse our sparsity. Normally n won't be bigger than 5.

Generating text with a n-gram Language Model

Profit *after financial* _______

Sample from the probability distribution:

word	probability
income	0.035
crisis	0.022
support	0.031
report	0.032
with	0.0000001
run	0.0000001
…	…

Try it yourself

Language Model

Language Modeling:

Input: sequence of words: $x_1, x_2, x_3 \dots x_n$

Output: *probability distribution* of the next word: $P(x_{n+1} \mid x_n, x_{n-1} \dots x_1)$

Neural Language Model

Neural **Language Modeling** runs on a fixed-window, like n-gram:

Input: fixed-window sequence of last k words:

$P(x_n \mid x_{n-1}, x_{n-2} \ldots x_{n-k}) = \mathit{softmax(MLP(x))}$

$X = \mathrm{encode}(x_{n-1}, x_{n-2} \ldots x_{n-k})$

How do we encode the text?

One-hot encoding

We have k elements in a vocabulary of size |V|

Let's assume k=4, |V| = 10 and we want to encode(x1, x2, x3, x4)

V={A,B,C,D,E,F,G,H,I,J}

One-hot-encoding of *k* elements

```
A=[1,0,0,0,0,0,0,0,0,0]
B=[0,1,0,0,0,0,0,0,0,0]
C=[0,0,1,0,0,0,0,0,0,0]
D=[0,0,0,1,0,0,0,0,0,0]
E=[0,0,0,0,1,0,0,0,0,0]
F=[0,0,0,0,0,1,0,0,0,0]
G=[0,0,0,0,0,0,1,0,0,0]
H=[0,0,0,0,0,0,0,1,0,0]
I=[0,0,0,0,0,0,0,0,1,0]
J=[0,0,0,0,0,0,0,0,0,1]
```

How should we encode (D, A, G, C) ?

One-hot-encoding of *k* elements

```
A=[1,0,0,0,0,0,0,0,0,0]
B=[0,1,0,0,0,0,0,0,0,0]
C=[0,0,1,0,0,0,0,0,0,0]
D=[0,0,0,1,0,0,0,0,0,0]
E=[0,0,0,0,1,0,0,0,0,0]
F=[0,0,0,0,0,1,0,0,0,0]
G=[0,0,0,0,0,0,1,0,0,0]
H=[0,0,0,0,0,0,0,1,0,0]
I=[0,0,0,0,0,0,0,0,1,0]
J=[0,0,0,0,0,0,0,0,0,1]
```

encode(D, A, G, C) = $V_D + V_A + V_G + V_C$
= [1, 0, 1, 0, 0, 0, 1, 0, 0, 0]

What does it miss?

One-hot-encoding of *k* elements

```
A=[1,0,0,0,0,0,0,0,0,0]
B=[0,1,0,0,0,0,0,0,0,0]
C=[0,0,1,0,0,0,0,0,0,0]
D=[0,0,0,1,0,0,0,0,0,0]
E=[0,0,0,0,1,0,0,0,0,0]
F=[0,0,0,0,0,1,0,0,0,0]
G=[0,0,0,0,0,0,1,0,0,0]
H=[0,0,0,0,0,0,0,1,0,0]
I=[0,0,0,0,0,0,0,0,1,0]
J=[0,0,0,0,0,0,0,0,0,1]
```

encode(D, A, G, C) = $V_D \bullet V_A \bullet V_G \bullet V_C$

=[0,0,0,1,0,0,0,0,0,0, 1,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,1,0,0,0, 0,0,1,0,0,0,0,0,0,0,0]

Neural Language Model

Neural **Language Modeling** runs on a fixed-window, like n-gram:

Input: fixed-window sequence of last k words:

$P(x_n \mid x_{n-1}, x_{n-2} \ldots x_{n-k}) = \mathit{softmax(MLP(x))}$

X = encode($x_{n-1}, x_{n-2} \ldots x_{n-k}$)

Neural Language Model

***Neural* Language Modeling** runs on a fixed-window, like n-gram:

Input: fixed-window sequence of last k words:

$P(x_n \mid x_{n-1}, x_{n-2} \ldots x_{n-k}) = softmax(MLP(x))$

X = encode($x_{n-1}, x_{n-2} \ldots x_{n-k}$)

$$MLP(x) = softmax(g(g(xW^1 + b^1)W^2 + b^{2)}W^3 + b^3)$$

Neural Language Model

***Neural* Language Modeling** runs on a fixed-window, like n-gram:

Input: fixed-window sequence of last k words:

$P(x_n \mid x_{n-1}, x_{n-2} \ldots x_{n-k}) = softmax(MLP(x))$

$X = \text{encode}(x_{n-1}, x_{n-2} \ldots x_{n-k})$

$$MLP(x) = softmax(g(g(xW^1 + b^1)W^2 + b^{2)}W^3 + b^3)$$

Aggregation Option:

[0,0,0,1,0,0,0,0,0,0]
+
[1,0,0,0,0,0,0,0,0,0]
+
[0,0,0,0,0,0,1,0,0,0]
+
[0,0,1,0,0,0,0,0,0,0]
=
[1,0,0,1,0,0,1,0,0,0]

W

A=	[-0.32, 0.09, 0.33,-0.44]
B=	[0.29, 0.02,-0.46,-0.39]
C=	[-0.46, 0.24,-0.16, 0.08]
D=	[-0.15,-0.31, 0.34, 0.00]
E=	[-0.10,-0.37, 0.01, 0.40]
F=	[-0.28,-0.26,-0.24, 0.31]
G=	[-0.32,-0.42,-0.21, 0.18]
H=	[-0.09,-0.01, 0.06, 0.14]
I=	[0.28,-0.02,-0.39, 0.12]
J=	[0.23,-0.22,-0.14, 0.28]

$(\mathbf{V_D} + \mathbf{V_A} + \mathbf{V_G} + \mathbf{V_C})\ W = \mathbf{V_D\ W + V_A\ W + V_G\ W + V_C W}$

Sum of rows in W

Each row corresponds to a certain vocabulary item

Contatination Option:

W

A=	[-0.32, 0.09, 0.33,-0.44]
B=	[0.29, 0.02,-0.46,-0.39]
C=	[-0.46, 0.24,-0.16, 0.08]
D=	[-0.15,-0.31, 0.34, 0.00]
E=	[-0.10,-0.37, 0.01, 0.40]
F=	[-0.28,-0.26,-0.24, 0.31]
G=	[-0.32,-0.42,-0.21, 0.18]
H=	[-0.09,-0.01, 0.06, 0.14]
I=	[0.28,-0.02,-0.39, 0.12]
J=	[0.23,-0.22,-0.14, 0.28]

$(V_D \bullet V_A \bullet V_G \bullet V_C)\ W = ?$

Contatination Option

$(V_D \bullet V_A \bullet V_G \bullet V_C)\, W = ?$

$$[0,0,0,1,0,0,0,0,0,0]$$
$$\circ$$
$$[1,0,0,0,0,0,0,0,0,0]$$
$$\circ$$
$$[0,0,0,0,0,0,1,0,0,0]$$
$$\circ$$
$$[0,0,1,0,0,0,0,0,0,0]$$
$$=$$
$$[0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0]$$

still sum of rows in W but W has 4x many rows

D(-3)= [-0.12,-0.24, 0.12,-0.34]
E(-3)= [-0.42,-0.21, 0.08, 0.40]
F(-3)= [0.20, 0.11,-0.31, 0.33]
G(-3)= [0.07,-0.05, 0.16, 0.23]
H(-3)= [0.28, 0.03, 0.22,-0.49]
I(-3)= [0.08, 0.39,-0.25, 0.27]
J(-3)= [0.10,-0.42,-0.37, 0.35]
A(-2)= [-0.00, 0.41, 0.19, 0.49]
B(-2)= [0.24, 0.48, 0.34,-0.42]
C(-2)= [-0.46, 0.22, 0.24,-0.21]
D(-2)= [-0.11,-0.48, 0.18,-0.22]
E(-2)= [-0.32, 0.10,-0.41,-0.43]
F(-2)= [0.32, 0.02,-0.22, 0.06]
G(-2)= [-0.31,-0.36, 0.09, 0.39]
H(-2)= [0.01,-0.22,-0.09,-0.15]
I(-2)= [0.01, 0.10,-0.16,-0.21]
J(-2)= [-0.24, 0.40,-0.34,-0.13]
A(-1)= [-0.23,-0.38, 0.02, 0.32]
B(-1)= [-0.34, 0.04,-0.18,-0.00]
C(-1)= [0.40,-0.02, 0.10,-0.16]
D(-1)= [0.13,-0.07,-0.19,-0.01]
E(-1)= [0.40, 0.27,-0.33, 0.36]
F(-1)= [0.04,-0.13,-0.43, 0.39]
G(-1)= [0.44, 0.38, 0.03,-0.39]
H(-1)= [0.41,-0.23, 0.33,-0.08]
I(-1)= [-0.50,-0.16,-0.42,-0.27]
J(-1)= [-0.15, 0.41, 0.46,-0.16]
A(+0)= [-0.11, 0.03, 0.20, 0.50]
B(+0)= [0.16,-0.34, 0.20,-0.21]
C(+0)= [0.05,-0.13,-0.23,-0.31]

- 1-hot times matrix: matrix row selection
- Sum of 1-hot times matrix: row selection + sum
- Concat of 1-hot: like using 1-hot from larger vocab

Embedding Layer

$$\text{encode}(D, A, G, C)$$

$$= \mathbf{E}_{[D]} \circ \mathbf{E}_{[A]} \circ \mathbf{E}_{[G]} \circ \mathbf{E}_{[C]}$$

[-0.15,-0.31, 0.34, 0.00,-0.32, 0.09, 0.33,-0.44,-0.32,-0.42,-0.21, 0.18,-0.46, 0.24,-0.16, 0.08]

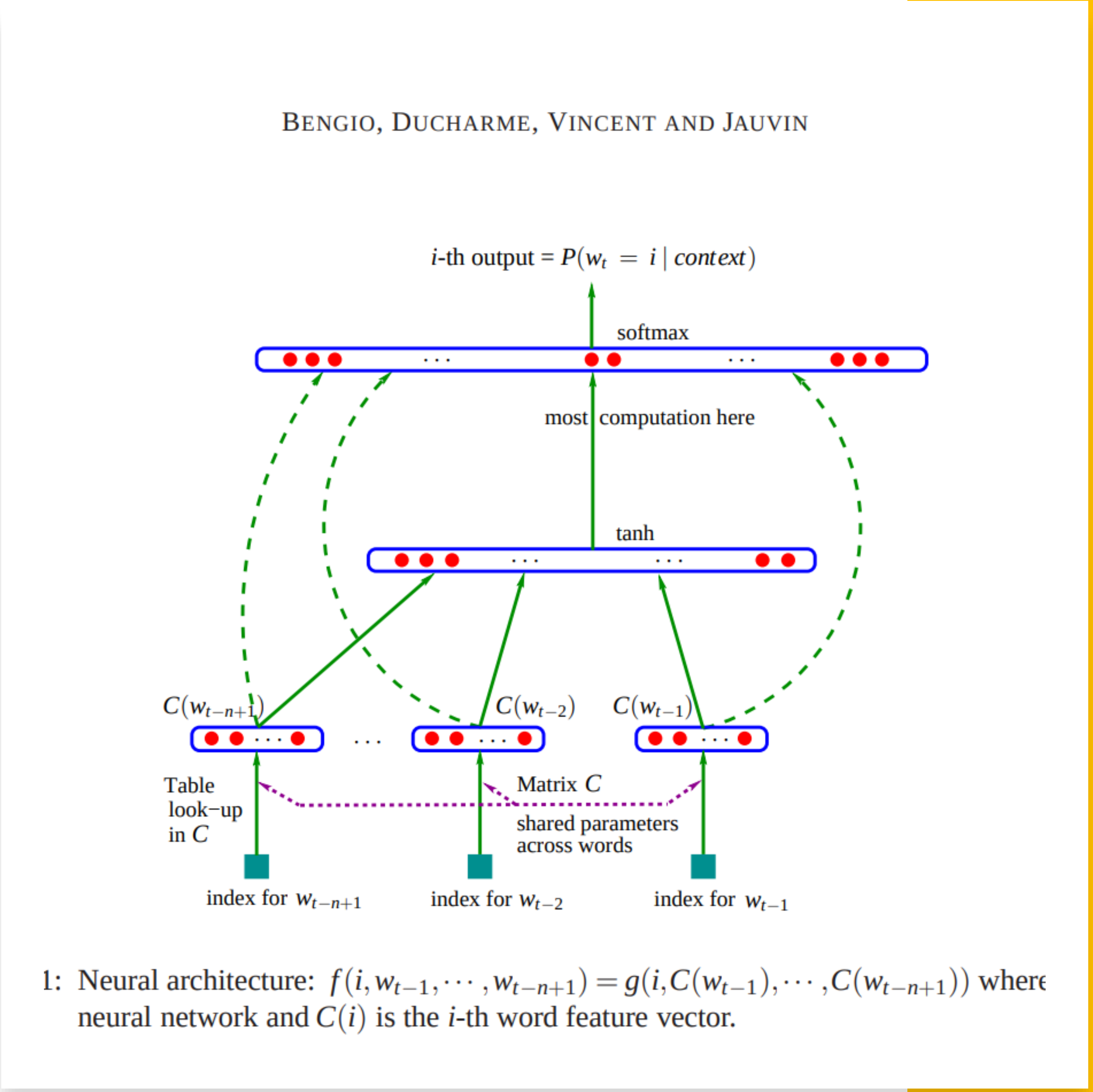
E

A= [-0.32, 0.09, 0.33,-0.44]
B= [0.29, 0.02,-0.46,-0.39]
C= [-0.46, 0.24,-0.16, 0.08]
D= [-0.15,-0.31, 0.34, 0.00]
E= [-0.10,-0.37, 0.01, 0.40]
F= [-0.28,-0.26,-0.24, 0.31]
G= [-0.32,-0.42,-0.21, 0.18]
H= [-0.09,-0.01, 0.06, 0.14]
I= [0.28,-0.02,-0.39, 0.12]
J= [0.23,-0.22,-0.14, 0.28]

- Assigns each item of the vocabulary a unique number
- Associate it with a row in matrix **E** of dense vectors (row dimension << |V|)
- concat or sum rows of **E** for the given input

A Neural Probabilistic Language Model

- Bengio et al. (2003)



Training a neural language model

- Set dimensions of the layers E, W3, according to your vocab size.
- Initialize with random values for E, W1, W2, W3, b1, b2, b3
- For every n-tuple in some text:
    - try to predict last item based on prev n-1
    - use cross-entropy loss

Neural Language Model can do:

- Probability score for a given sentene
- Generate new sentences
- Predict next word *i* based on previous *k* words
- Predict word label based on *k* items (when?)

What happens after the training?

Consider the columns of the last layer W3.

Consider the rows of the embedding layer E.

The columns of W3 corresponds to the vocabulary items (!)

Review

Note the most important thing you've learnt so far

The issue with one-hot-vector

```
motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
```

Issues with the previous method

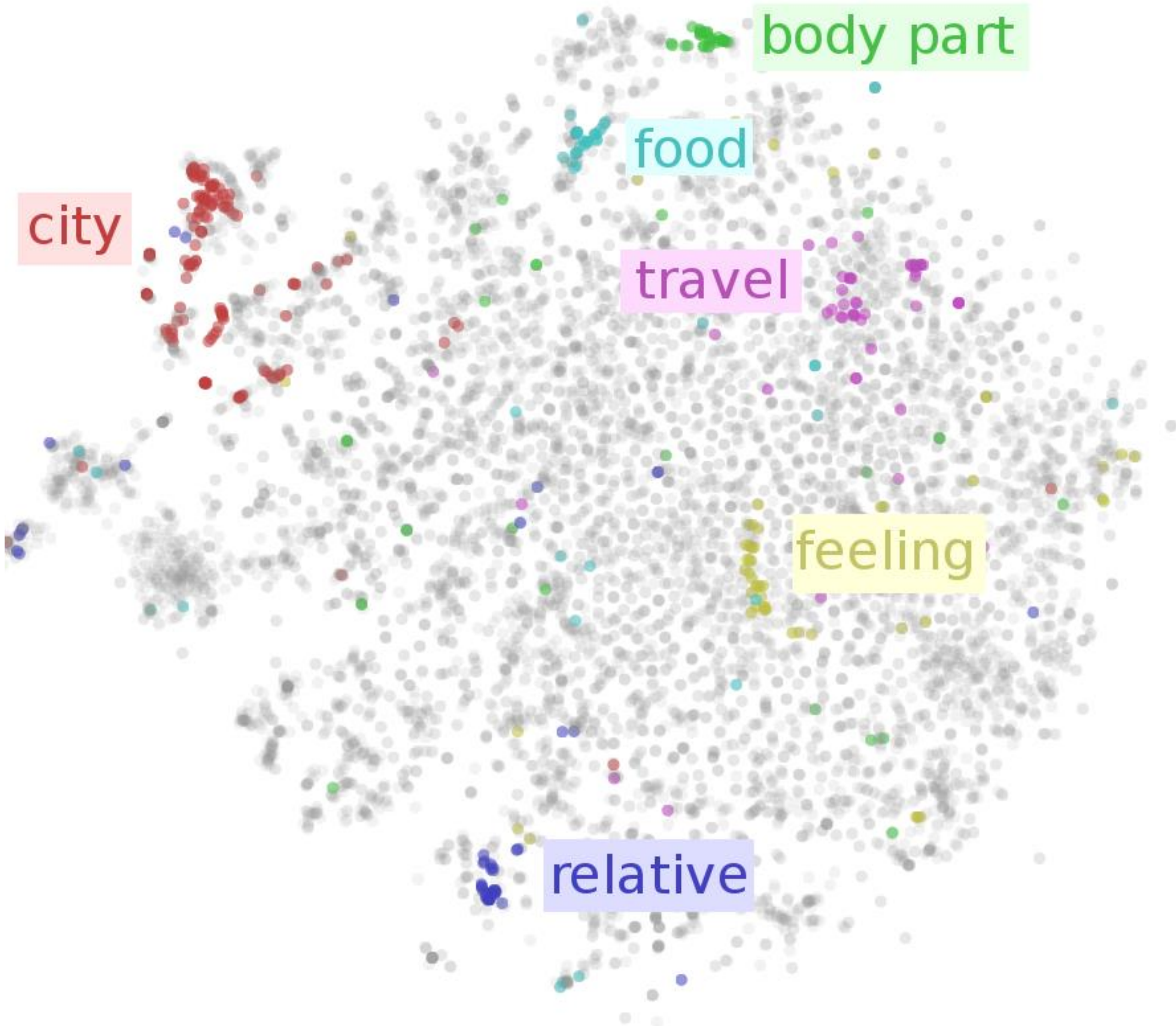
```
motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
```

- Words are treated as discrete symbols: no sense of similarity.
  The vectors are orthogonal
- Vector Dimension = # of words in the vocabulary.
- Training a language model is expensive (why?)
- We want **better vector representation**

How?

Word Vectors

- Aka:
  Word Embedding /
  Word Representations /
  Distributed Representation



Your Turn!

Word2Vec Family Algorithm

You have **1h** to prepare short presentation about **Distributional Semantics**:

- *Group A*: Word2Vec
- *Group C*: Neural Word Embedding as Implicit Matrix Factorization
- *Group D*: gloVe
- *Group E*: FastText

Prepare to teach the rest:

- Read The Papers
- Read additional Material
- Prepare a presentation (~20 min long)
- Points to Cover:
  - Model structure
  - Training process
  - Output vectors semantic properties
  - Differences (I.e., from the language model)
  - Critics

Distributional Semantics

A word’s meaning is given by the words that frequently appear with it.

“You shall know a word by the company it keeps”
(J. R. Firth 1957: 11).

When a word *w* appears in a text, its context is the set of words that appear nearby (within a fixed-size window).

Neural Language Model can use this context to build a represntation of w.

A fixed-window neural Language Model

We need a neural network that can operate on *variable lengthes* of input

Improvements over n-gram LM:

- No sparsity problem
- Don’t need to store all observed n-grams

Remaining problems:

- Fixed window is too small
- Enlarging window makes W bigger
- Window can never be large enough!
- Every word is multiplied by **completely different weights** in W.
  No symmetry in how the inputs are processed.

Language Models – Recommended Reading

- https://lilianweng.github.io/lil-log/2019/01/31/generalized-language-models.html