

# Language Models & Word2vec - Summary

---

Liad Magen



# Let's Review:



We used a fixed-sized window  
for classification (ex-5)



We learned about  
Distributional Semantics

# A fixed-window Neural Language Model

We need a neural network  
that can operate on  
*variable lengths* 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 the 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.

# 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).



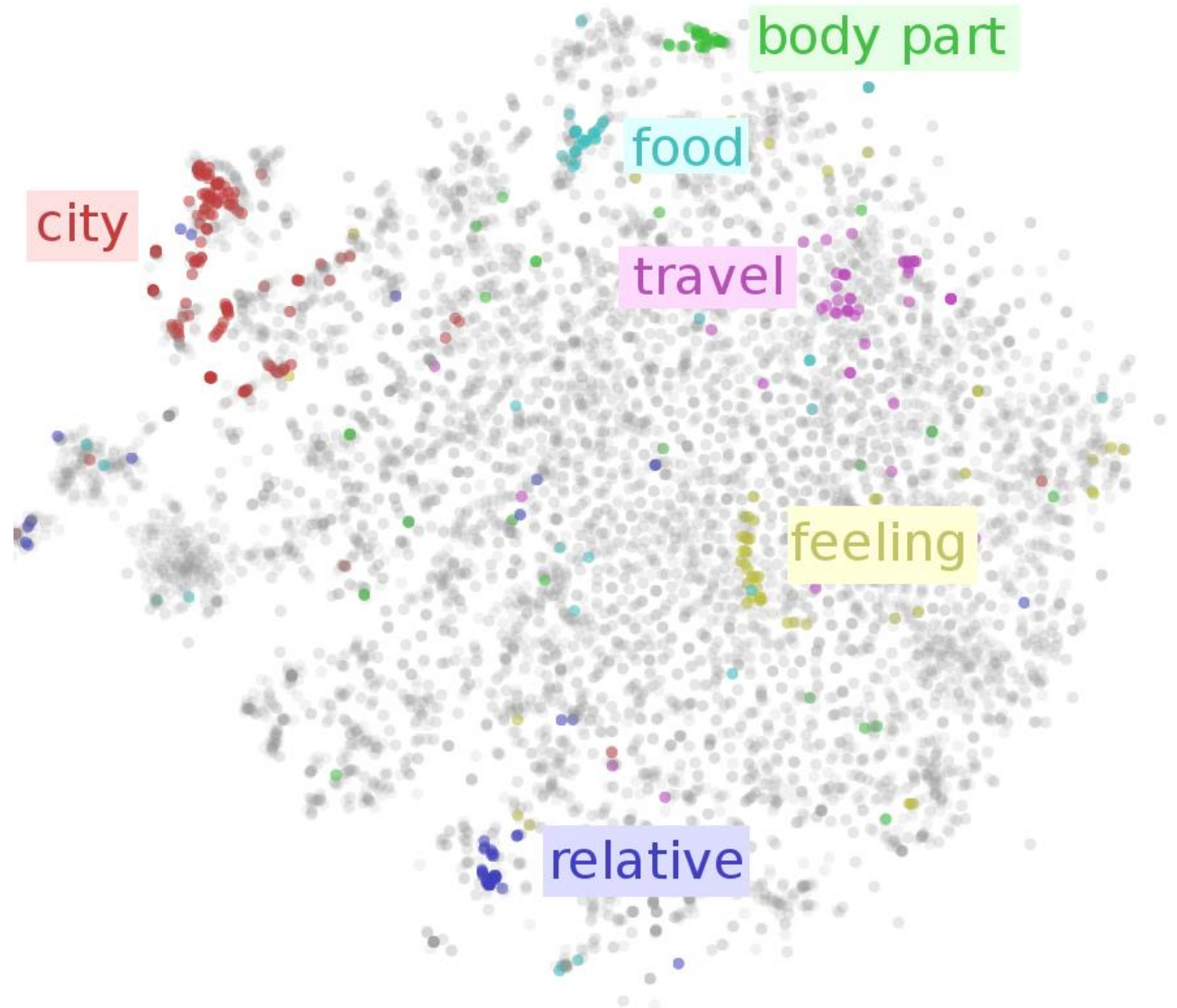
When a word  $w$  appears in a text, its context is the set of words that appear nearby (within a fixed-size window).



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

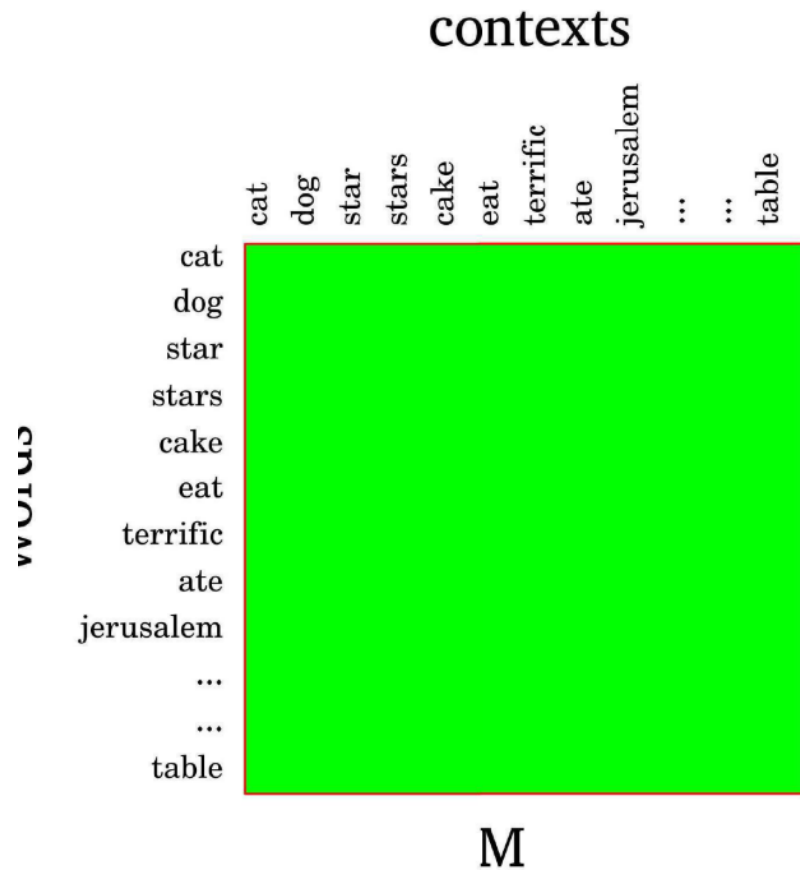
# Word Vectors

- Aka: *Word Embedding, Word Representations, Distributed Representation*



# Neural Language Models

- (Neural) Language Model:
  - **Probability distribution** over word combination
  - Results a useful **word vectors representations**
  - Have aspects of word similarity
- But - expensive to train



# Words as vectors

- We can arrange words in a *huge*, sparse matrix, where each row is a word, and each column is a context.
- Apply Dimensionality Reduction:
  - Singular Value Decomposition
  - Non-negative matrix factorization
  - Latent Dirichlet Allocation (LDA) ...

# Solution: Word2Vec algorithm

- We only care about the co-occurrences
- Matrix Decomposition
- Fast to train
  - No layers
  - No softmax
- 2 Versions:
  - CBOW
  - Skip-Grams



# Or: Co-occurrence count

- Another option is to count words (or lemmas) co-occurrences:
  - In a document
  - In a sliding window
- Also called 1<sup>st</sup>-order similarity
- Great for topic-modeling (clustering);  
Leads to: **Latent Semantic Analysis** (LSA)
- Related topic: **Collocations**. See: [colloc.pdf \(stanford.edu\)](#)



# Review: Word2Vec algorithm

- Init a network with random word vectors
- Iterate through each word in the whole corpus
- Try to predict surrounding words using the word vectors
- Update the vectors so to improve predictions
- This algorithm learns word vectors which capture *word similarity* and *meaningful directions* in a word-space



Similarity?

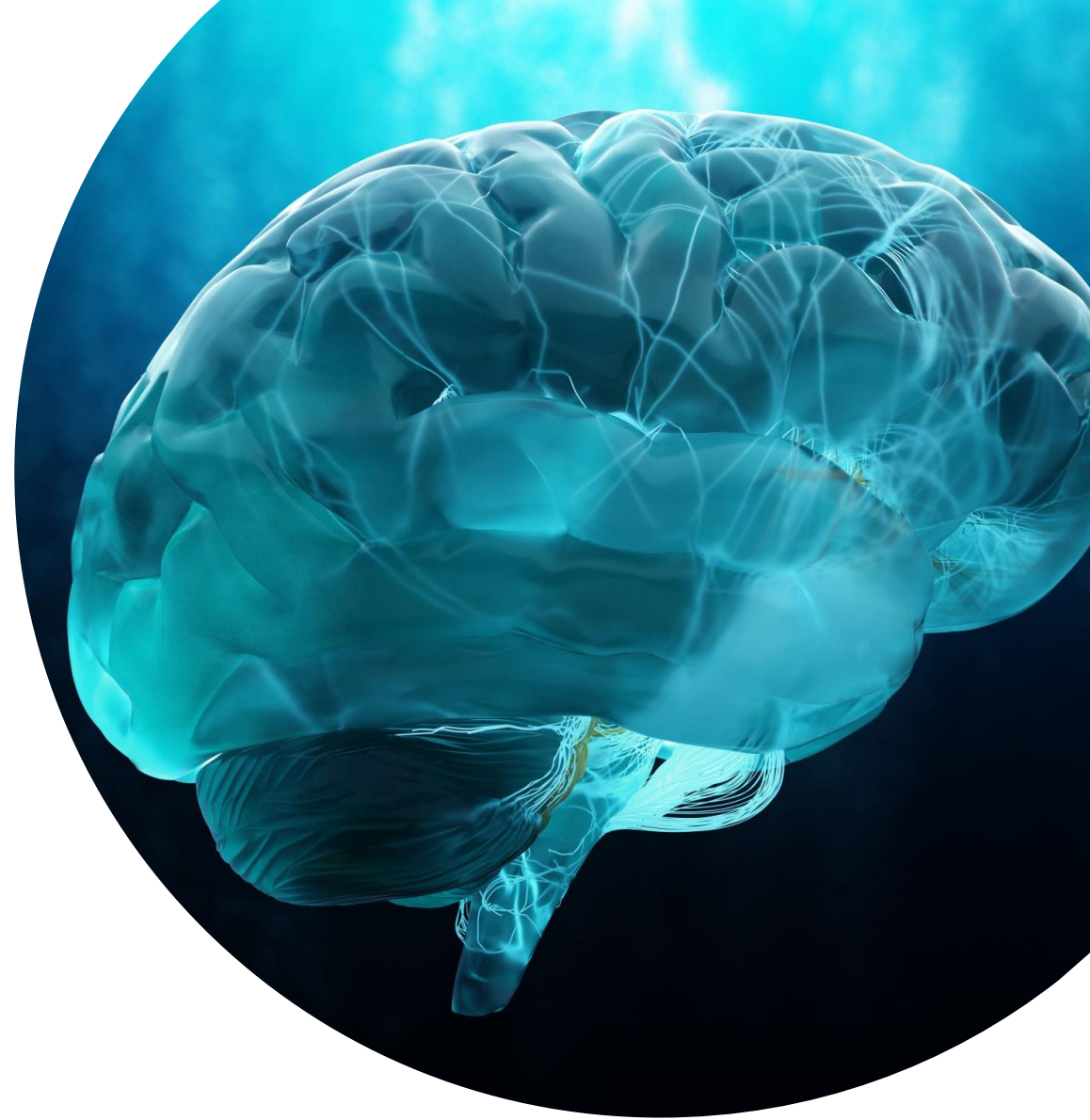


# Distributional Representation of Word Meaning and Similarity

What kind of meaning?

# What kind of meaning?

- We don't know what "meaning" is
  - Some neural representation in the brain?
- We will focus on representations that can predict meaning similarities



# Understanding the Words

soup was bad  
soup was awful  
soup was lousy  
soup was abysmal  
soup was icky

chowder was nasty  
pudding was terrible  
cake was bad  
hamburger was lousy

service was poor  
atmosphere was shoddy  
hammer was heavy

- To computers, each word is just a symbol, so these are all the same.
- But to us some are more similar than others.
- We'd like a word representation that can capture that.

# Distributional Hypothesis

“Because the capsule was hermetically broamed, its contents were in perfect condition after more than a hundred years underwater.”

[Testing the Distributional Hypothesis: The influence of Context on Judgements of Semantic Similarity \(escholarship.org\)](#)



# Working with Dense Vectors

---

## Word Similarity

- Similarity is calculated using *cosine similarity*

$$\text{similarity}(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n A_i^2} \times \sqrt{\sum_{i=1}^n B_i^2}}$$

- When the vectors are normalized ( $\|A\| = 1$ ) – it's equivalent to a dot product.
- You can normalize the vectors after loading them

# Working with Dense Vectors

- Finding the most similar words to the vector  $v$ 
  - Compute the similarity from word  $v$  to all the other words
  - This is a single matrix-vector product:  $W \cdot v^T$
  - The result is a  $|V|$  sized vector of similarities
- Take indices of the top  $k$  values

$$\begin{array}{c}
 \begin{array}{c} d \\ |V| \\ \begin{array}{c} \text{cat} \\ \text{chair} \\ \text{june} \\ \text{sun} \\ \text{bark} \\ \dots \\ \dots \\ \text{eat} \end{array} \end{array} \\
 W \\ |V| \times d
 \end{array}
 \begin{array}{c}
 \begin{array}{c} \text{dog} \end{array} \\
 v^T \\ d \times 1
 \end{array}
 =
 \begin{array}{c}
 \begin{array}{cccccccc} 0.9 & -0.3 & -0.1 & -0.9 & 0.3 & \dots & \dots & 0.2 \end{array} \\
 \begin{array}{c} \text{cat} \\ \text{chair} \\ \text{june} \\ \text{sun} \\ \text{bark} \\ \vdots \\ \vdots \\ \text{eat} \end{array} \\
 \text{similarities} \\ 1 \times |V|
 \end{array}$$





# Similarity to **several** words

- Calculate the pairwise similarities and sum them:
  - $W \cdot \text{manager} + W \cdot \text{business} + W \cdot \text{quit} =$   
 $W \cdot (\text{manager} + \text{business} + \text{quit})$
- Now find the indices of the highest values as before.



# Pre-trained embeddings

- Key success of NLP
- Semi-supervised learning (+ transfer learning).
- Fine-tuning = some features change, while most stay the same.



# Pre-training

- Train the network on a correlated task
- Take feature representations as an input to another model

# Pre-training

- In *word2vec*, the tasks were:
  - CBOW: "predict word based on a window of size  $k$  around it"
  - Skip-Gram: "predict neighboring words in a window of  $k$  around a focus word"
- More generally:  
"predict word based on some **context** of the word".

- **Useful**: we can get tons of training data for free.
- The choice of **contexts** determines the **resulting word representations**

# Choosing a context

## Window of $k$ around a word.

- smaller  $k$ : more syntactic
- Larger  $k$ : more semantic
- Time of the current message
- User who wrote the message
- ... Sky is the limit

## Aligned words in a different language

- Requires a parallel corpus (synonyms, paraphrases)

# Solutions for “infinite” vocabularies

- FastText

Represent a word by the sum of its char n-grams:

dinosaur = dinosa + inosau + nosaur + dino + inos + nosa + osau + saur + din + ino + nos + osa + sau + aur

- Word Pieces, BPE:

dinosaur = dino #sa #ur

- Characters:

dinosaur = d i n o s a u r

- When is it better?

- When is it worse?

The background of the slide features a complex network diagram with nodes and edges, overlaid with various mathematical symbols and equations. Visible symbols include  $\pm$ ,  $\%$ ,  $\Omega^\pm$ ,  $\times$ ,  $e^{i\pi}$ ,  $\&$ ,  $1=0$ ,  $\infty$ ,  $>$ ,  $\frac{1}{4}$ ,  $F-E+V=2$ ,  $\nabla \times H = \frac{\partial D}{\partial t}$ ,  $\nabla \cdot B \approx 0$ ,  $\div$ ,  $\gamma$ ,  $\Delta$ ,  $\frac{\partial^2 u}{\partial t^2} = c^2$ ,  $E = mc^2$ , and  $\%00$ .

# Dealing with Sequences

Sentence = many word vectors

Document = even more word vectors

What can we do when using Neural Network?

- Sum/Avg all the sentence vectors
- Concat: Fixed-window input (size n)



# Sum/Avg word vectors

- Continuous Bag Of Words:  $CBOW(f_1, \dots, f_k) = \frac{1}{k} \sum_{i=1}^k v(f_i)$
- CBOW works surprisingly well for classification
- Can assign weights to each feature:  $WCBOW(f_1, \dots, f_k) = \frac{1}{\sum_{i=1}^k a_i} \sum_{i=1}^k a_i v(f_i)$
- Can even capture word-order (+-)
  - Or - adding some **k** as order:

$$k = 1 \rightarrow \frac{1}{2}, k = 2 \rightarrow \frac{1}{3}, k = 3 \rightarrow \frac{1}{4}, \dots, k = \frac{1}{k+1}$$

[Deep Unordered Composition Rivals Syntactic Methods for Text Classification - ACL Anthology](#)

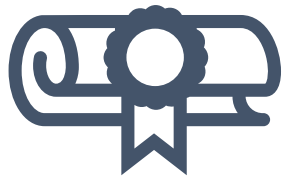


# Software

- [CIMeC Wiki | CLIC \(unitn.it\)](#)
  - Given vector representation of words, derive vector representation of phrases/sentences
  - Implements various composition methods



# Try the word vectors out yourself.



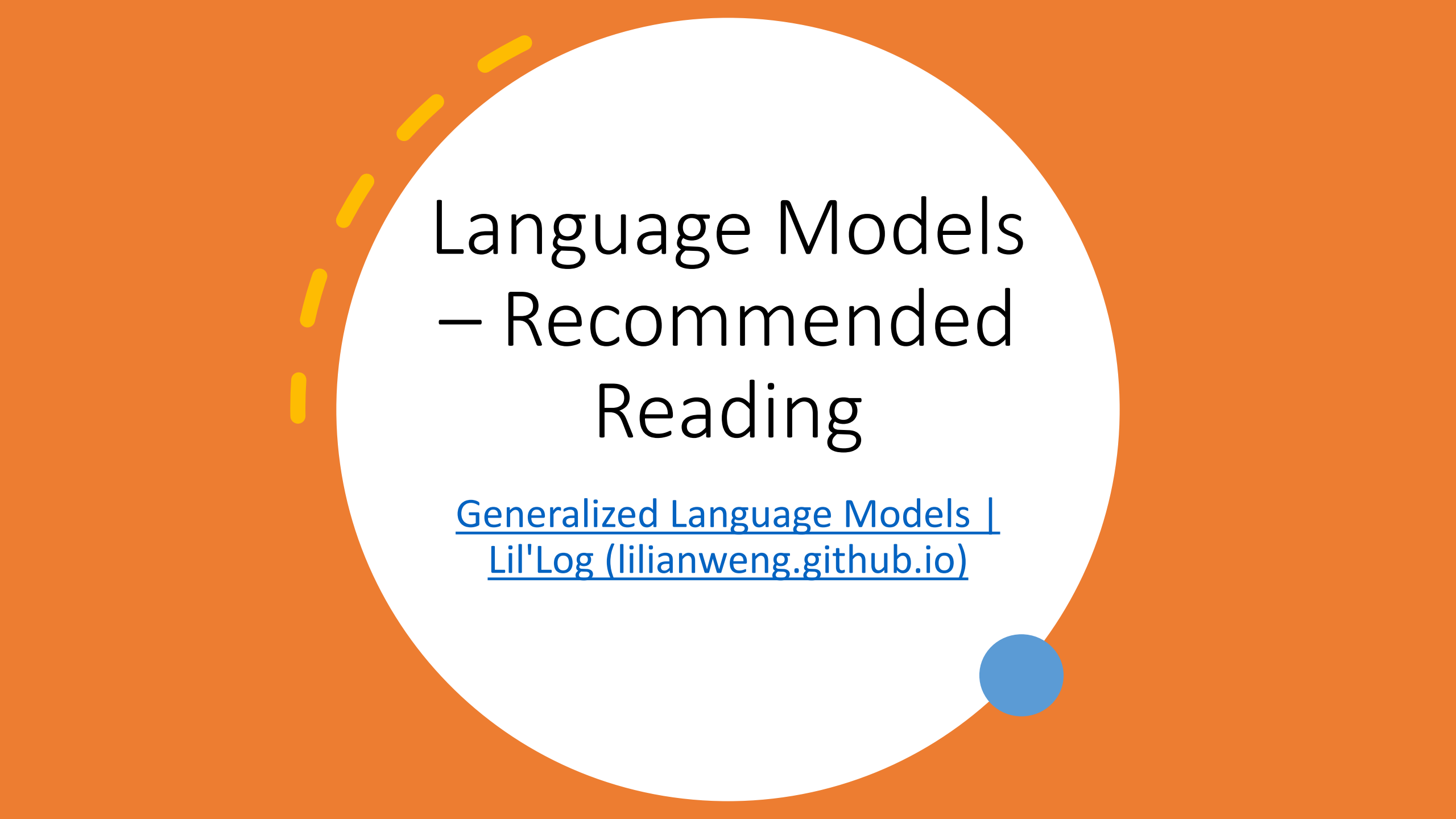
## Exercise 07

Optional – but worth reading

Demonstrates the level in universities



## Exercise 08 – playground



# Language Models – Recommended Reading

[Generalized Language Models |  
Lil'Log \(lilianweng.github.io\)](#)