A crash course in language models

Sebastian Schuster
Saarland University

March 20, 2023

Slides available at: TODO

Plan for today

- What are language models?
- One type of neural language models (NLMs):
  Recurrent NLMs
- Using them for linguistic research
    - As tools
    - To test hypotheses about human language processing/learning

What is a language model?

P(*next word* | *context*)

A conditional probability distribution over the **next word** from a fixed vocabulary,
given **a sequence of previous words**.

What is a language model?

P(*next word* | "The cat")

Next word	**P(next word | context)**
a	0.0000006
aardvark	0.000002
aarhus	0.0000001
…	
mat	0.0000003
…	
on	0.004
…	
sat	0.1
…	
zebra	0.00007

Scoring words and sequences

Scoring words:

$$P(\,\textit{next word} \mid \textit{context}\,)$$

Scoring sequences:

$$P(\,\textit{on a mat} \mid \textit{the cat sat}\,)$$

$$= P(\,\textit{on} \mid \textit{the cat sat}\,)$$

Generating texts

the cat

Next word	**P(next word $\mid$ the cat)**
a	0.0000006
aardvark	0.000002
aarhus	0.0000001
…	
mat	0.0000003
…	
on	0.004
…	
sat	0.1
…	
zebra	0.00007

Generating texts

the cat sat

Next word	P(next word | the cat)
a	0.0000006
aardvark	0.000002
aarhus	0.0000001
…	
mat	0.0000003
…	
on	0.004
…	
sat	0.1
…	
zebra	0.00007

Generating texts

the cat sat

Next word	P(next word | the cat sat)
a	0.0000006
aardvark	0.000002
aarhus	0.0000001
…	
mat	0.0000003
…	
on	0.15
…	
sat	0.0001
…	
zebra	0.00007

Generating texts

the cat sat on

Next word	P(next word | the cat sat on)
a	0.2
aardvark	0.000002
aarhus	0.0000001
…	
mat	0.0000003
…	
on	0.0000015
…	
sat	0.0001
…	
zebra	0.00007

Generating texts

the cat sat on a

Next word	P(next word | the cat sat on a)
a	0.000004
aardvark	0.000002
aarhus	0.0000001
…	
mat	0.1
…	
on	0.0000015
…	
sat	0.0001
…	
zebra	0.007

Generating texts

the cat sat on a mat

Where do the probabilities come from?

- **Pre-2015ish:**
  - **Counting** short sequences in large corpora
  - One problem: Estimates are very poor for very rare sequences/sequences that don’t appear in the corpus
- **Post-2015ish:**
  - **Neural language models**

A neural language model

Context

Context of previous words $w_1, w_2, \ldots, w_k$

A neural network

$P\left(w_{k+1}\right)$

Probability distribution over the next word $P\left(w_{k+1}\right)$

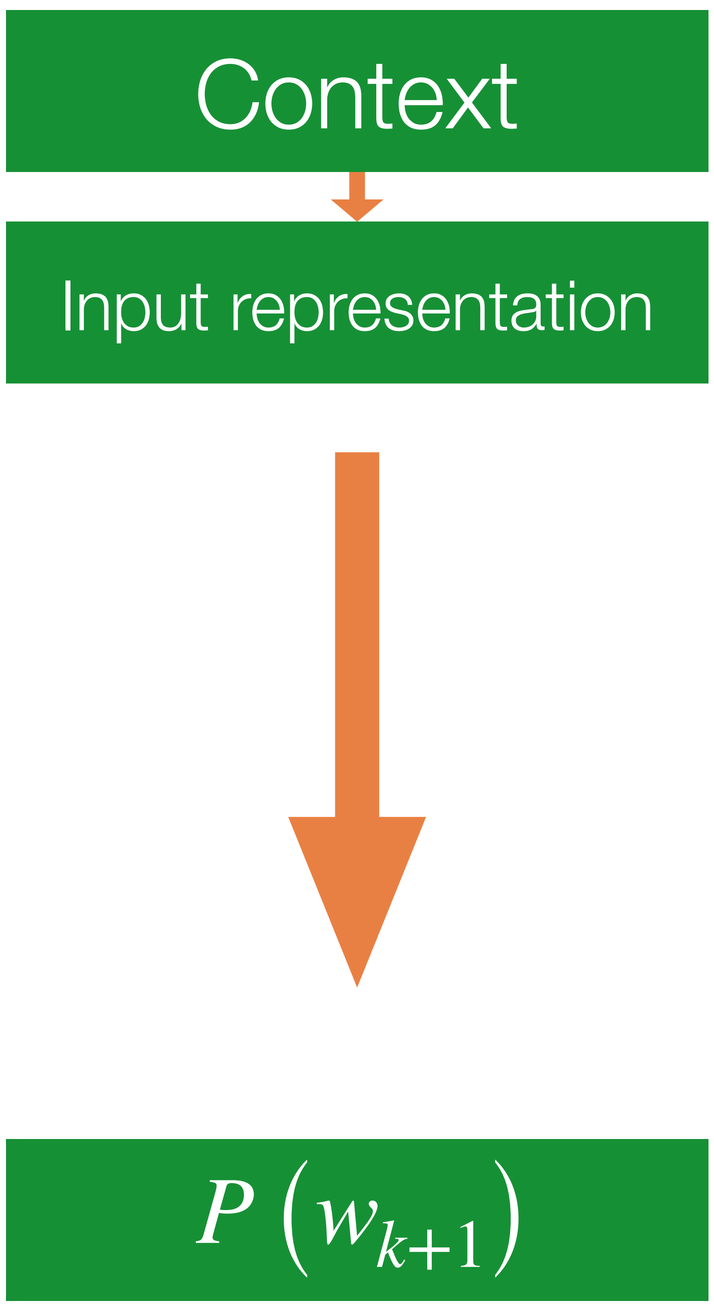
Representing words

- Neural networks can only operate on numbers
- We therefore represent words as **high-dimensional vectors**
- Vectors for each word are stored in **a lookup table** (the *embedding matrix*)

$$\begin{pmatrix} 0.544 \\ -0.678 \\ 0.604 \\ 0.944 \\ 0.632 \end{pmatrix} \quad \begin{pmatrix} -0.023 \\ 1.354 \\ -0.553 \\ -0.367 \\ 0.975 \end{pmatrix} \quad \begin{pmatrix} -1.079 \\ -0.612 \\ 0.594 \\ -1.057 \\ -1.186 \end{pmatrix} \quad \begin{pmatrix} -0.262 \\ -0.923 \\ 1.097 \\ -0.724 \\ -1.078 \end{pmatrix} \quad \begin{pmatrix} 0.352 \\ -0.341 \\ 0.318 \\ 0.345 \\ -1.452 \end{pmatrix}$$

the cat sat on a

A neural language model



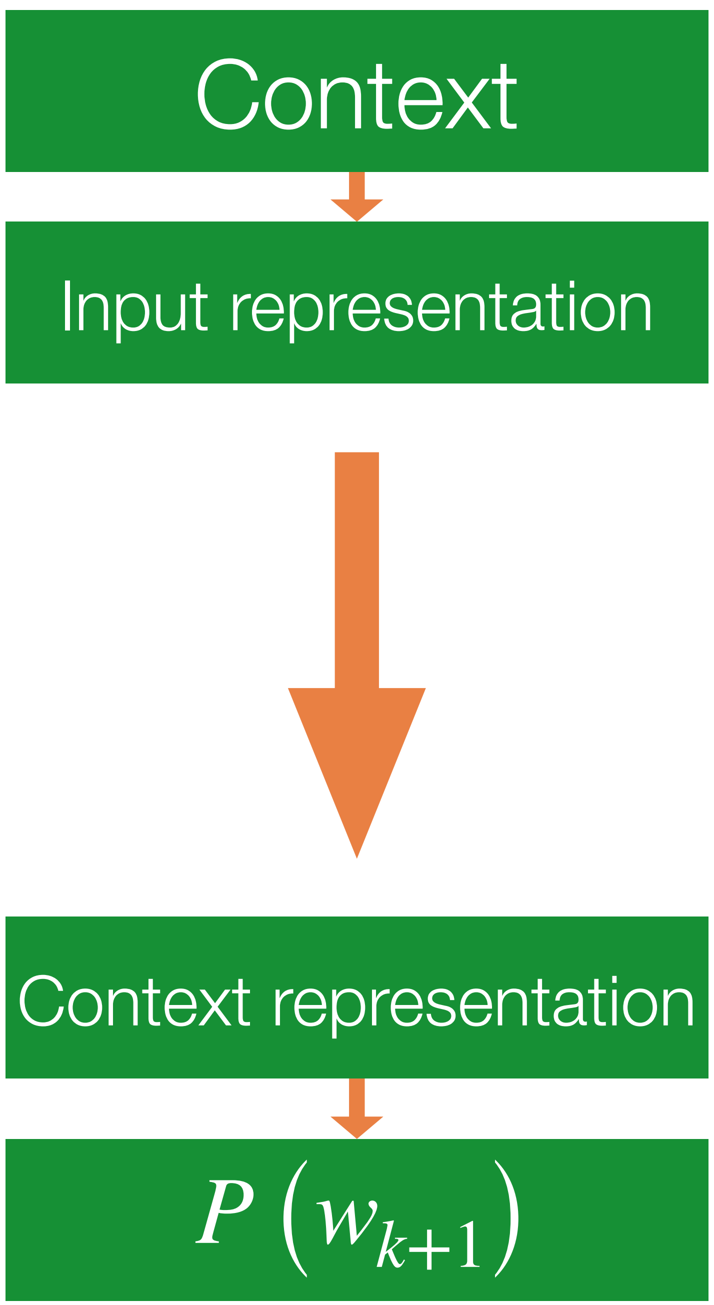
Context of previous words $w_1, w_2, \ldots, w_k$

Word vectors

A neural network

Probability distribution over the next word $P\left(w_{k+1}\right)$

A neural language model



Context of previous words $w_1, w_2, \ldots, w_k$

Word vectors

A neural network

Vector representing the context

Probability distribution over the next word $P\left(w_{k+1}\right)$

Computing the probability of the next word: SoftMax

Context representation

$\downarrow$

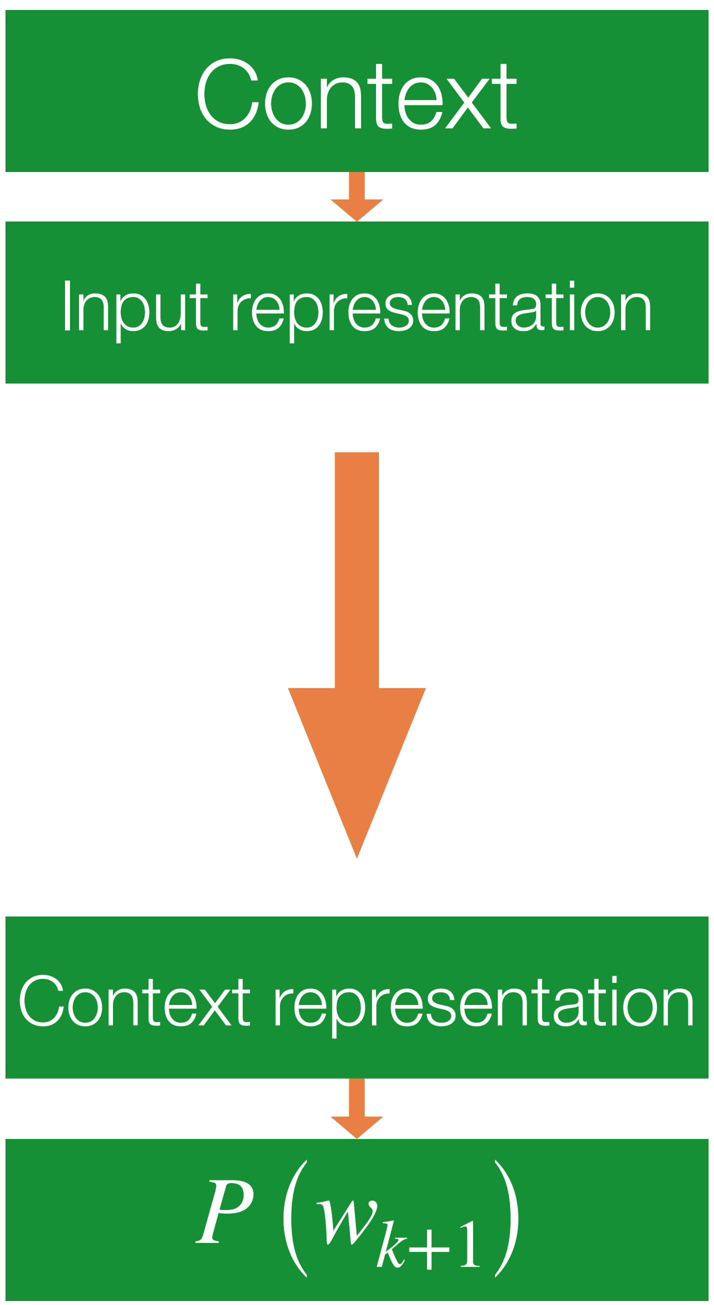
$P\left(w_{k+1}\right)$

l-dimensional vector representing the context c

Probability distribution over the next word $P\left(w_{k+1}\right)$

$P(w_{k+1} \mid c) =$ Multinomial logistic regression (aka SoftMax) classifier using features c

A neural language model



Context of previous words $w_1, w_2, \ldots, w_k$

Matrix with word embeddings

A neural network

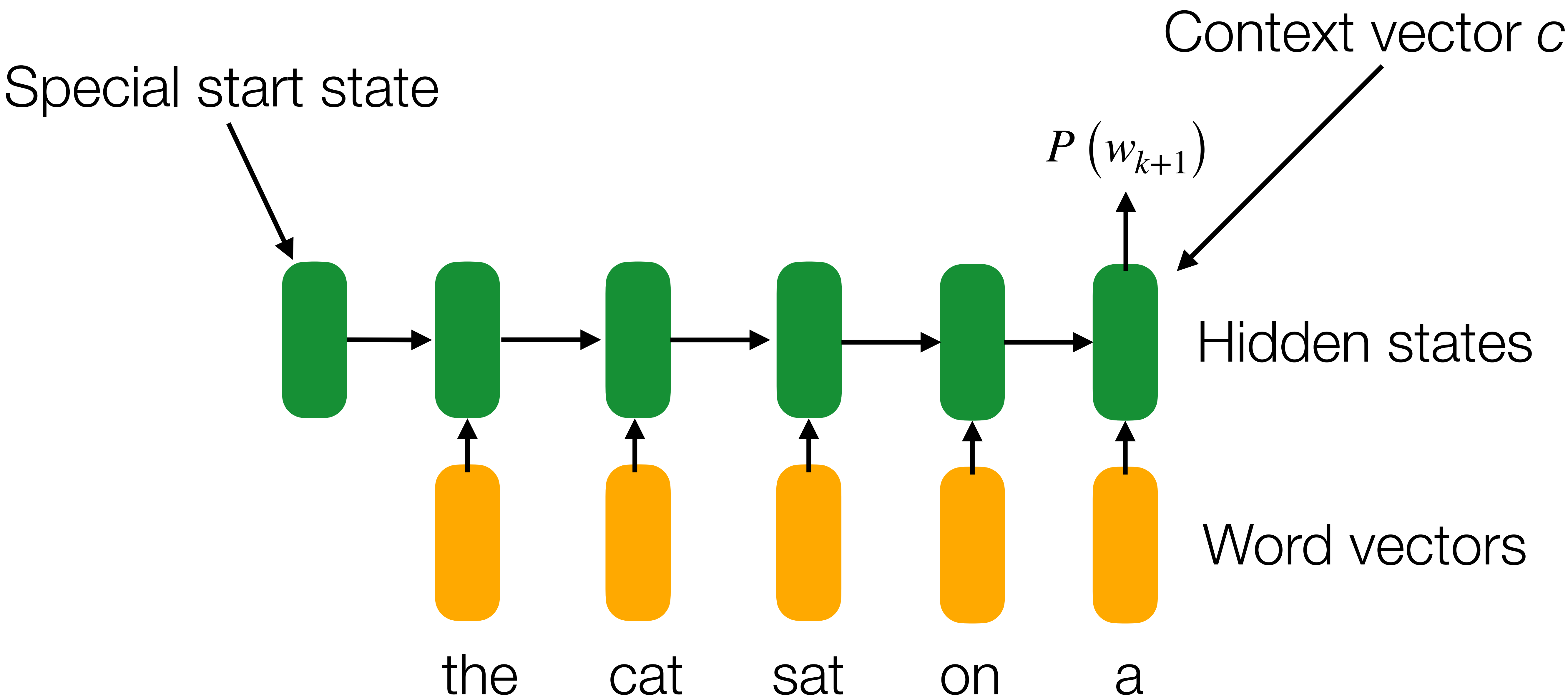
Vector c representing the context

Probability distribution over the next word $P\left(w_{k+1}\right)$

Two methods to compute context representations

- **Transformers:**
  - Central idea: Repeatedly compute **weighted averages** of word representations
- **Recurrent Neural Networks (RNNs):**
  - Central idea: **Combine current word with a hidden state** representing the previous words

Recurrent Neural Networks



A neural language model

Context	Context of previous words $w_1, w_2, \ldots, w_k$
↓	
Input representation	Matrix with word embeddings
↓	
e.g., RNNs	Recurrent combination of word vectors and representation of previous context
↓	
Context representation	Vector c representing the context
↓	
$P\left(w_{k+1}\right)$	Probability distribution over the next word $P\left(w_{k+1}\right)$

Training neural networks: Backpropagation

- A neural network has many **parameters** (e.g., for composition function, word embedding matrix, …)
- During training the network **makes guesses** about the next word
- If the guess is far off, the **parameters are updated** so that if the network guesses again, it is more likely to make the correct guess
- This process is repeated **millions or billions of times**

Pre-trained neural networks

- Medium-scale:
  - Available in Python on HuggingFace
- Large-scale (e.g, GPT-3/4, ChatGPT, Claude):
  - Accessible in the Cloud

How to use them?

- As **tools** for research tasks:
    - For helping with stimuli generation
    - Developing classifiers, e.g., for social media analysis
- For **testing hypotheses about language processing/ learning:**
    - To test hypotheses about online
    - To test hypotheses about learnal

For stimuli generation

B

I have a hypothesis that adjective ordering preferences within noun phrases are not as strong in Spanish as they are in English. Can you design some English and Spanish stimuli that would help me test that hypothesis in a controlled between-subjects psycholinguistics study involving two populations, Spanish speakers and English speakers? Please give me the stimuli in a consistent JSON format.

Here are some sample stimuli that could be used to test your hypothesis. These stimuli could be presented to participants in a controlled between-subjects study, with one group consisting of Spanish speakers and the other group consisting of English speakers. The stimuli are presented in a JSON format for consistency.

Copy code

```
{
  "study_name": "Adjective Ordering Preferences in Noun Phrases",
  "stimuli": [
    {
      "language": "English",
      "sentence": "The big red ball"
    },
    {
      "language": "English",
      "sentence": "The red big ball"
    },
    {
      "language": "Spanish",
      "sentence": "La pelota grande roja"
```

h/t Brandon Waldon

Automatic classification: Social media analysis

- Example classification task: code-switching (yes/no)

  1. Hand-label a small dataset

  2. Fine-tune a classifier using a pre-trained language model

  3. Automatically predict labels for the rest of the data

Testing hypotheses about incremental processing

COGNITIVE SCIENCE
A Multidisciplinary Journal

Cognitive Science 45 (2021) e12988

ISSN: 1551-6709 online
DOI: 10.1111/cogs.12988

Single-Stage Prediction Models Do Not Explain the Magnitude of Syntactic Disambiguation Difficulty

Marten van Schijndel, PhD,[a] Tal Linzen, PhD[b]

[a]*Department of Linguistics, Cornell University*
[b]*Department of Linguistics and Center for Data Science, New York University*

Received 26 August 2020; received in revised form 21 April 2021; accepted 26 April 2021

van Schijndel and Linzen (2021)

Some things to consider when using LMs as cognitive models

- **Amount of training data:** Recent models are trained on orders of magnitude more linguistic data than a human receives an input over the course of their life

- **Autoregressive vs. bidirectional language models:** There are also popular language models that predict a word based on the left and right context — not a good model for left-to-right online processing!

- **Tokenization:** Most models split up words into smaller units (so-called subword tokens). Many of th e not linguistically meaningful.

Some things to consider when using LMs as cognitive models

- **Memory constraints**: Transformer-based models have perfect memory of hundreds, or even thousands of words

- **Resist anthropomorphizing**: Interacting with models such as ChatGPT can feel sometimes similar to interacting with humans — this doesn't mean the models produce responses like a human would.

- …

Takeaways

- Language models are **conditional probability distributions over the next word, given a context**

- Neural networks constitute a powerful method for learning such a distribution

- Language models can be used **as tools for research tasks**, and if well justified, for **testing hypotheses about human language processing**

    - More about that in my talk tomorrow!

Thanks!

- **Additional resources:**
  - Jurafksy & Martin: Speech and Language Processing (3rd ed)
  - Giuliano Giacaglia: How Transformers Work
  - Sasha Rush: The Annotated Transformer
  - HuggingFace Model Hub
  - Stanford CS224N Lectures

Transformers: Self-attention

- **Intuition:** the output representation y_i of a word w_i should be a combination of **its own representation** and the representations of **other words that it depends on** (syntactically, in terms of meaning, …)

- We do this by computing an **attention vector** α_i

- The output representation y_i is a **weighted sum** of all the input representations

$$y_i = \sum_{0 \le j \le k} \alpha_{ij} w_j$$

Transformers

- Instead of computing these weighted averages just once, Transformer models usually consist of multiple layers (in practice, usually somewhere between 5 and 20 layers)

- The input of layer l is the output of layer $l - 1$