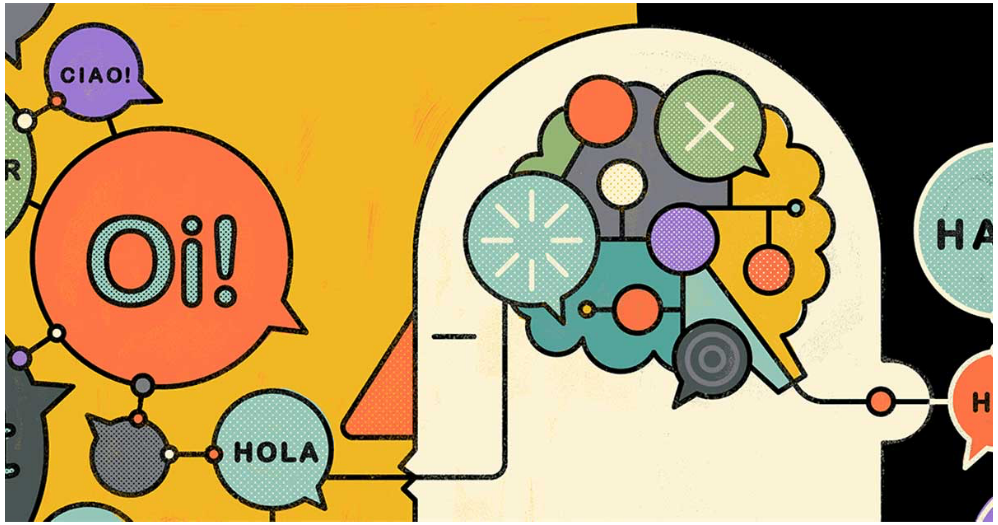
Department of Linguistics and Translation
香港城市大學
City University of Hong Kong

Language and its Applications LT5903



Jixing Li

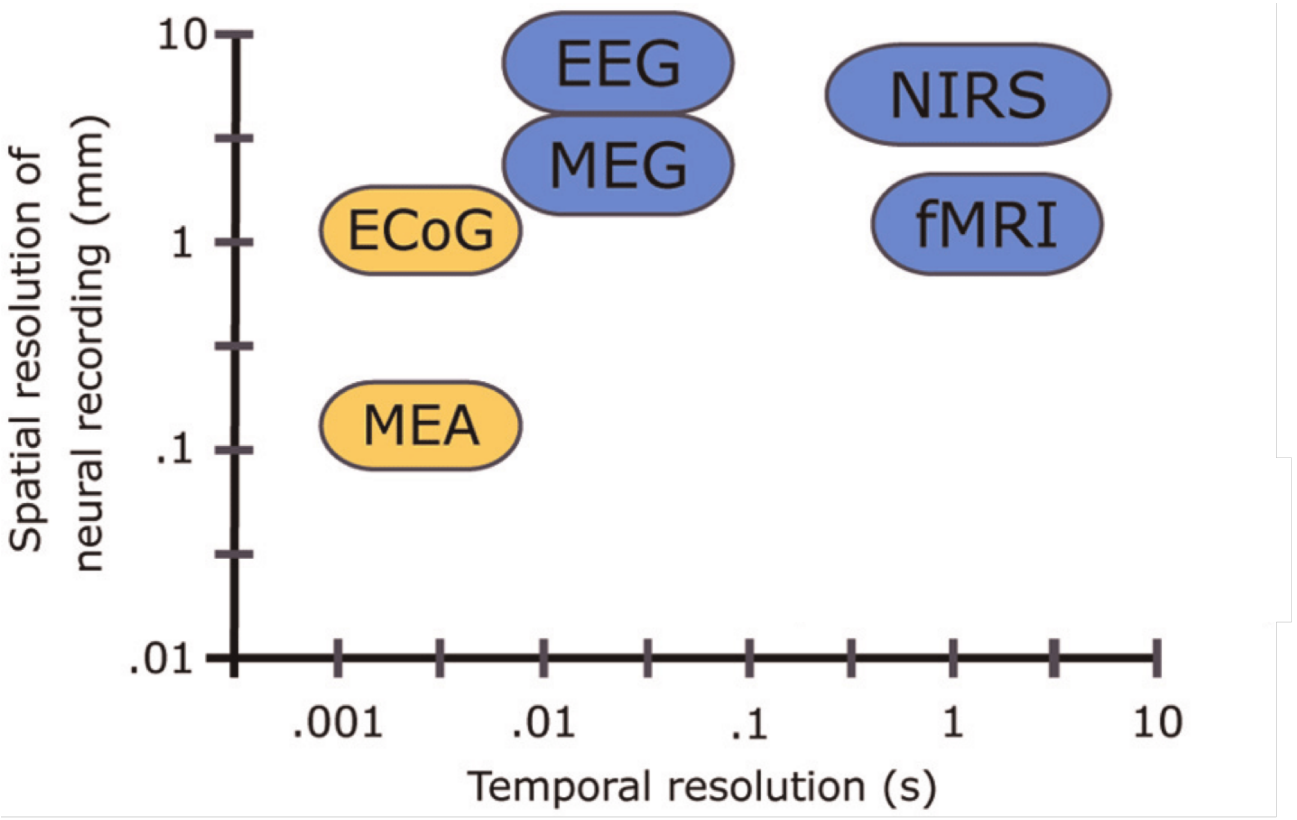
Lecture 10: Computational Linguistics

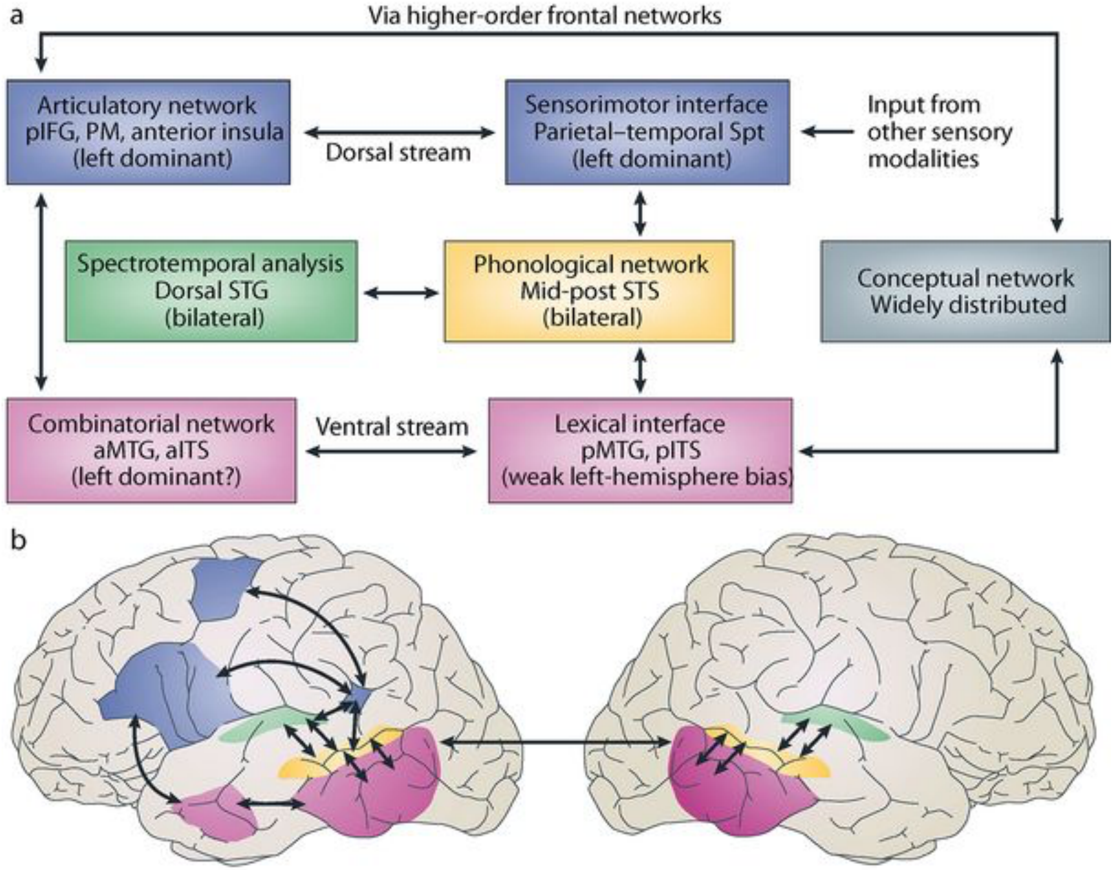
Lecture plan

- Neurolinguistics review
- Computational linguistics and natural language processing
- Tokenization, POS tagging
- CFG and parsing
- Word embeddings
- Short break (15 mins)
- Group discussion on HW10

Neurolinguistics review

neurolinguistics: the study of how language is represented in the **brain**
research methods: EEG, MEG, ECoG, fMRI, etc





Computational linguistics

computational linguistics (CL): employs computational methods to understand properties of human language.

natural language processing (NLP): aims to develop methods for solving practical problems involving language

NLP tasks: information extraction, automatic speech recognition, machine translation, sentiment analysis, question answering, and summarization.

Every NLP task requires …

- Tokenizing (segmenting) words

```
word_tokenize("Computational Linguistics is fun!")
```

```
['Computational', 'Linguistics', 'is', 'fun', '!']
```

- Normalizing word format   e.g., lower case, remove punctuation

```
['computational', 'linguistics', 'is', 'fun']
```

- Segmenting sentences

```
sent_tokenize('Computational Linguistics is fun! Tokenization is easy.')
```

```
['Computational Linguistics is fun!', 'Tokenization is easy.']
```

Example: Getting web pages

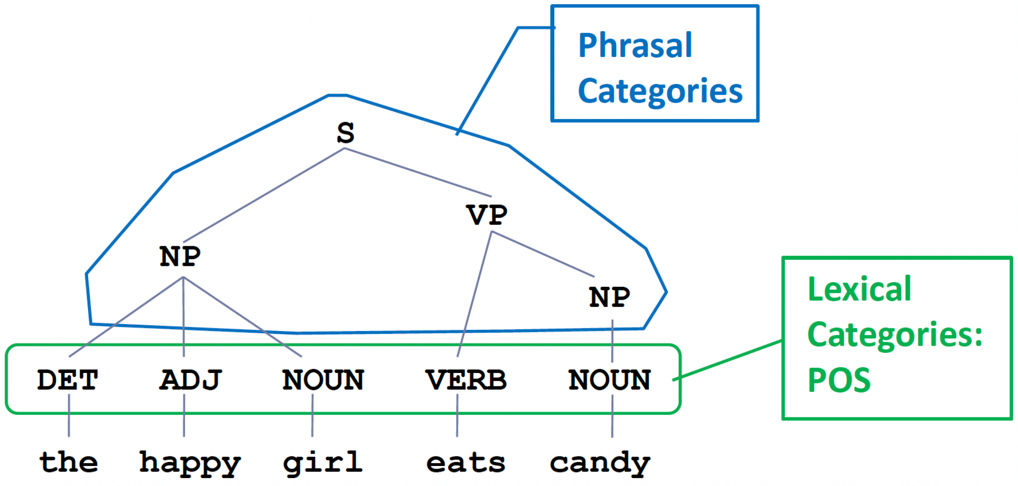
```
# get webpage
html = urlopen('https://www.hplovecraft.com/writings/texts/fiction/cc.aspx')
# get raw text
raw = BeautifulSoup(html).get_text()
# find the index where the relevant text starts
ind_start = re.search('“Of such great powers', raw).start()
raw = raw[ind_start:]
# tokenization
tokens = word_tokenize(raw)
# remove punctuation
tokens = [t for t in tokens if t.isalpha()]
tokens_lower = [t.lower() for t in tokens]
# show the first 20 tokens
tokens_lower[:20]
```

Part-of-speech (POS) tagging

- POS tagging: assign a POS tag to each word, symbol, punctuations in a sentence.

```
sent = word_tokenize('I saw a man with binoculars')
tokens_pos = nltk.pos_tag(sent, tagset='universal')
tokens_pos
```

```
[('I', 'PRON'),
 ('saw', 'VERB'),
 ('a', 'DET'),
 ('man', 'NOUN'),
 ('with', 'ADP'),
 ('binoculars', 'NOUN')]
```



@Jixing Li

Context-free grammars

Context-free grammars (CFG) are also called **Phrase-Structure Grammars.** The idea of basing a grammar on constituent structures is formalized by Chomsky (1956)

A CFG consists of **a set of rules** and **a lexicon** of words and symbols

start symbol

S → NP VP
NP → DT N
VP → V NP
DT → the
V → robbed
N → burglar | apartment

non-terminal symbols

terminal symbols

alternate possible expansions

@Jixing Li

Top-down parsing

CFG:	**Input:**	**Stack**	**Operation**	
S → NP VP	*'the dog laughs'*	S	**expand**	S → NP VP
NP → DT N	*'the dog laughs'*	NP VP	**expand**	NP → DT N
DT → the	*'the dog laughs'*	DT N VP	**expand**	DT → the
N → dog	*'~~the~~ dog laughs'*	~~the~~ N VP	**scan**	the
VP → V	*'dog laughs'*	N VP	**expand**	N → dog
V → laughs	*'~~dog~~ laughs'*	~~dog~~ VP	**scan**	dog
	'laughs'	VP	**expand**	VP → V
	'laughs'	V	**expand**	V → laughs
	~~'laughs'~~	~~laughs~~	**scan**	laughs
	[]	[]		

@Jixing Li

Bottom-up parsing

CFG:	**Input:**	**Stack**	**Operation**	
S → NP VP	*‘the dog laughs’*	the	**shift**	the
NP → DT N	*‘dog laughs’*	DT	**reduce**	DT → the
DT → the	*‘dog laughs’*	DT dog	**shift**	dog
N → dog	*‘laughs’*	DT N	**reduce**	N → dog
VP → V	*‘laughs’*	NP	**reduce**	NP → DT N
V → laughs	*‘laughs’*	NP laughs	**shift**	laughs
	[]	NP V	**reduce**	V → laughs
	[]	NP VP	**reduce**	VP → V
	[]	S	**reduce**	S → NP VP

Word meaning: Attributes

Binder et al. (2016): 65 dimensions, scale: 0-6

Word	Vision	Bright	Dark	Color	Pattern	Large	Small
ant	3.5484	0.3548	3.5806	3.9355	1.9355	0.0968	5.871
bicycle	5.3	1.1667	0.6333	1	2.1667	1.7	1.2667
farm	5.7097	1.1935	0.5161	1.7419	1.8065	5.0645	0.129
farmer	4.1786	0.5	0.3214	0.4286	0.6071	1.4286	0.6786
green	4.2963	1.7778	1	5.9259	1.5926	0.1852	0.1111
red	5	3.2857	1.25	6	1.4643	0.1071	0.0357
rocket	5.5	2.9333	0.7333	1.8667	1.9	5.6	0.2333
trust	0.3793	0.1379	0.0345	0.3103	0.2069	0.3103	0.069

Word meaning: Co-occurrence

Wittgenstein (1953): The meaning of a word is its use in the language

Harris (1954): If A and B have almost identical environments we say that they are synonyms.

Firth (1957): A word is characterized by the company it keeps.

Example: *ongchoi*

Suppose you see these sentences:

ongchoi is delicious sautéed with garlic.
ongchoi is superb over rice
ongchoi leaves with salty sauces

And you've also seen these:

…spinach sautéed with garlic over rice
chard stems and leaves are delicious
collard greens and other salty leafy greens

Conclusion:

ongchoi is a leafy green like spinach, chard, or collard greens

We could conclude this based on words like "leaves" and "delicious" and "sauteed"

Word2Vec: skip-gram training

Assume a +/- 2 word window, given training sentence:

…lemon, a [tablespoon of apricot jam, a] pinch…

c1 c2 c3 c4

Goal: train a classifier that is given a candidate (word, context) pair

(apricot, jam)

(apricot, aardvark)

…

Assigns each pair a **probability**:

P(+|w, c): *c* is in the context of word *w*

P(−|w, c) = 1 − P(+|w, c)

Example

...lemon, a [tablespoon of apricot jam, a] pinch...

c1 c2 c3 c4

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

t	c	t	c
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

For each positive example we'll take *k* negative examples (here, *k*=2)

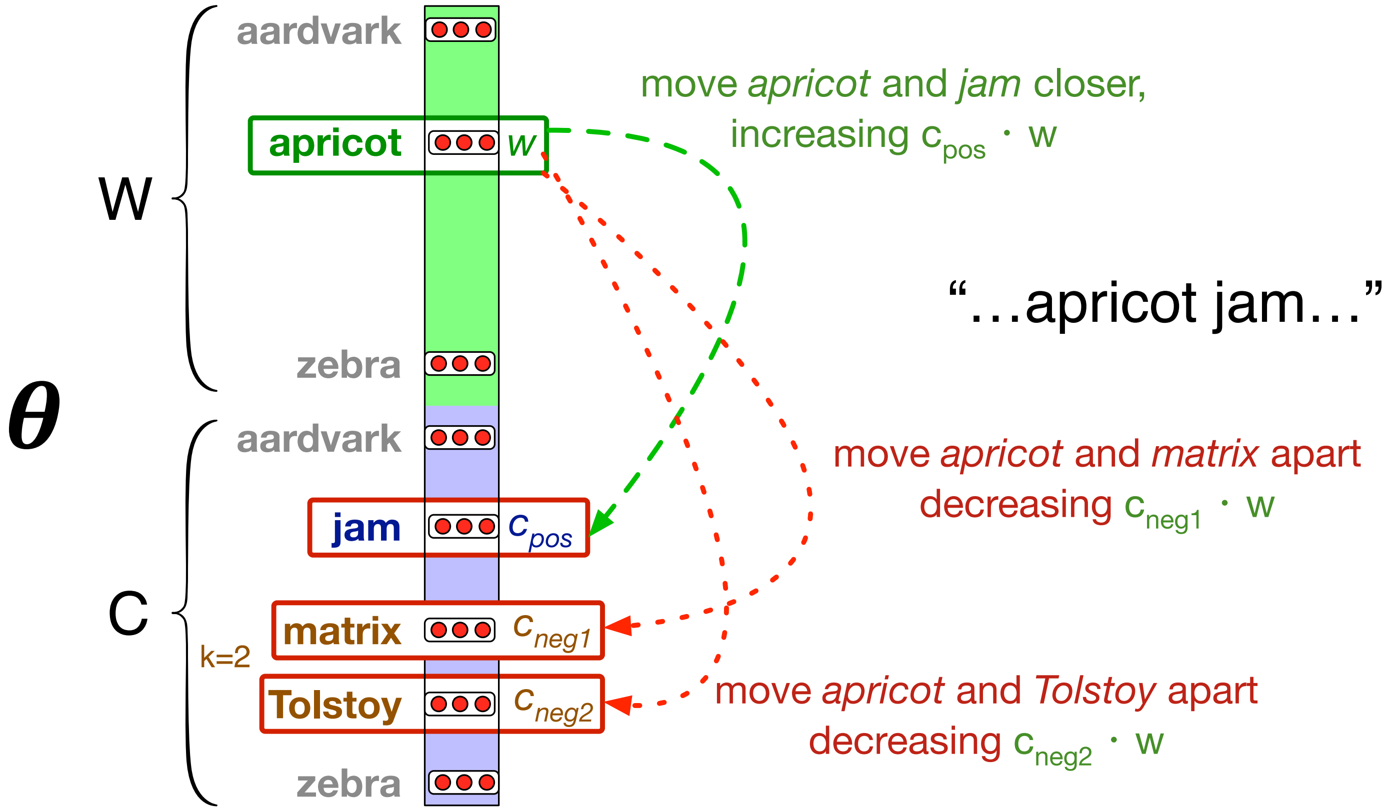
Learning the classifier

How to learn?

Gradient descent!

We'll adjust the word weights to

- make the positive pairs more likely
- and the negative pairs less likely,
- over the entire training set.



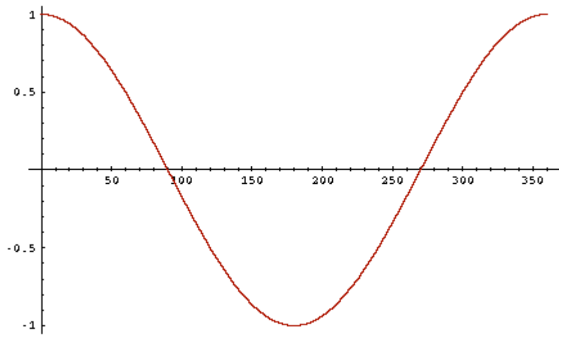
Computing word similarity: Cosine

$$\text{cosine}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

$$= v_1 w_1 + v_2 w_2 + ... + v_N w_N$$

Normalized by the length of the vector

The dot product tends to be high when the two vectors have large values in the same dimensions
→ a useful similarity metric between vectors



-1: vectors point in opposite directions: dissimilar
+1: vectors point in same directions: similar
0: vectors are orthogonal

Cosine similarity: Example

$$\cos\left(\frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|}\right) = \frac{\vec{v}}{|\vec{v}|}\cdot\frac{\vec{w}}{|\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

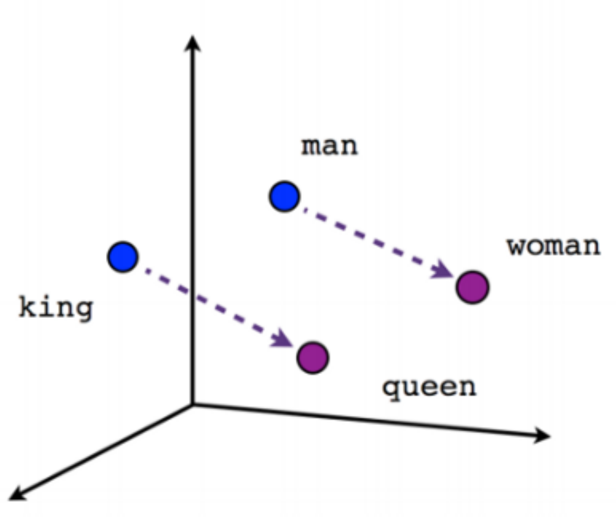
cherry	6.76	2.42	1.22
digital	1.65	6.85	6.83
information	1.44	6.62	6.48

$$\cos(cherry, information) = \frac{6.76 * 1.44 + 2.42 * 6.62 + 1.22 * 6.48}{\sqrt{6.76^2 + 2.42^2 + 1.22^2}\sqrt{1.44^2 + 6.62^2 + 6.48^2}} = 0.49$$

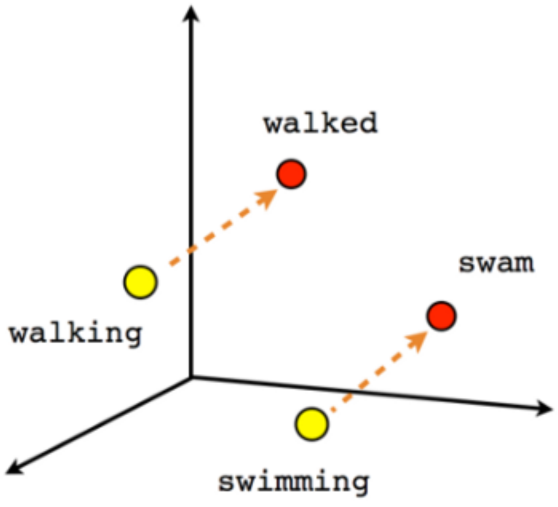
$$\cos(digital, information) = \frac{1.65 * 1.44 + 6.85 * 6.62 + 6.83 * 6.48}{\sqrt{1.65^2 + 6.85^2 + 6.83^2}\sqrt{1.44^2 + 6.62^2 + 6.48^2}} = 0.99$$

semantically-related words have higher cosine similarity

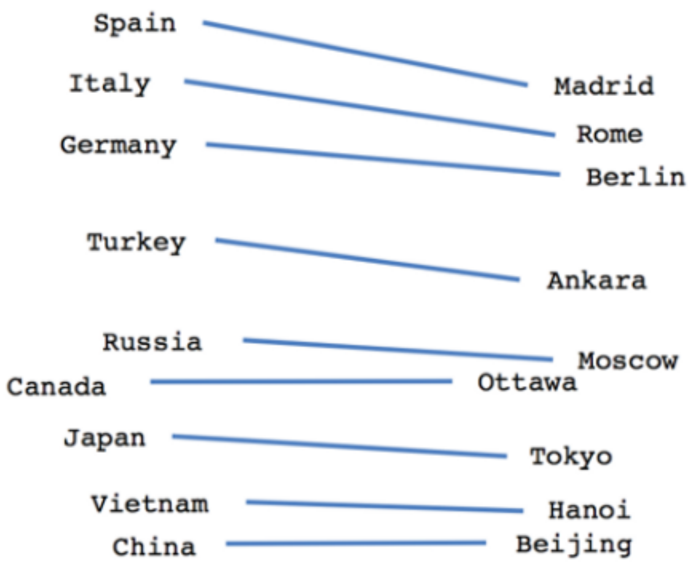
Evaluating word embeddings



Male-Female



Verb tense



Country-Capital

@Jixing Li

Against human judgement

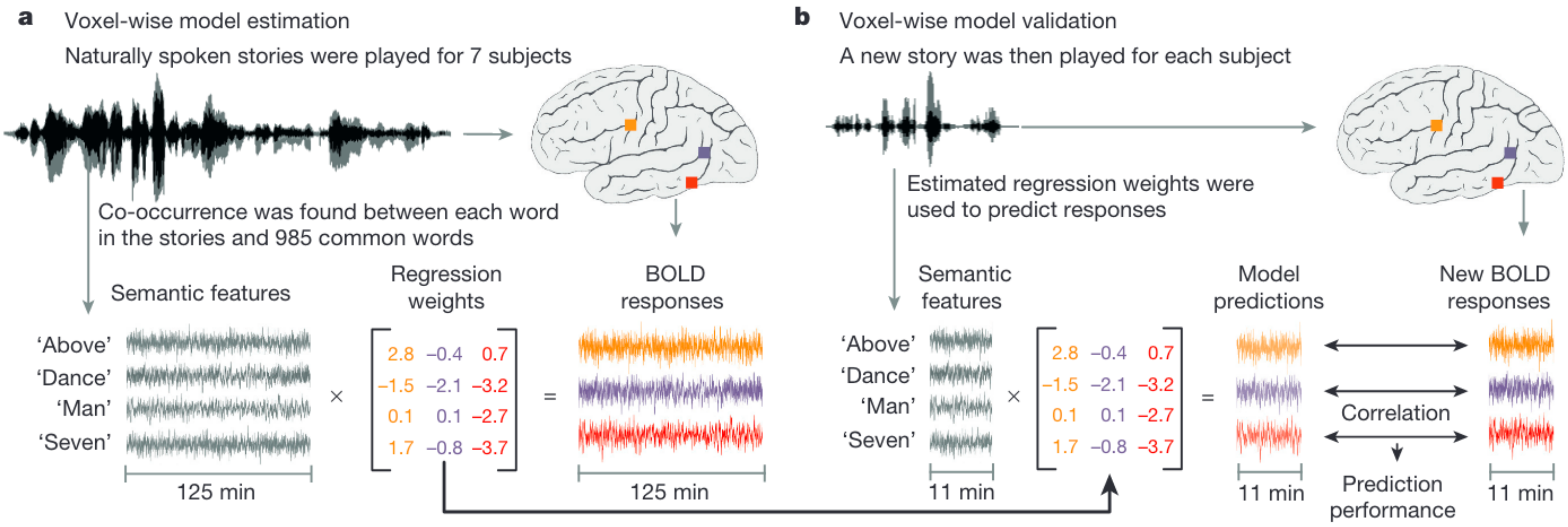
SimLex-999: Human rating on the similarity between 999 pairs of words (scale: 0-10)

word1	word2	similarity
vanish	disappear	9.8
behave	obey	7.3
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3

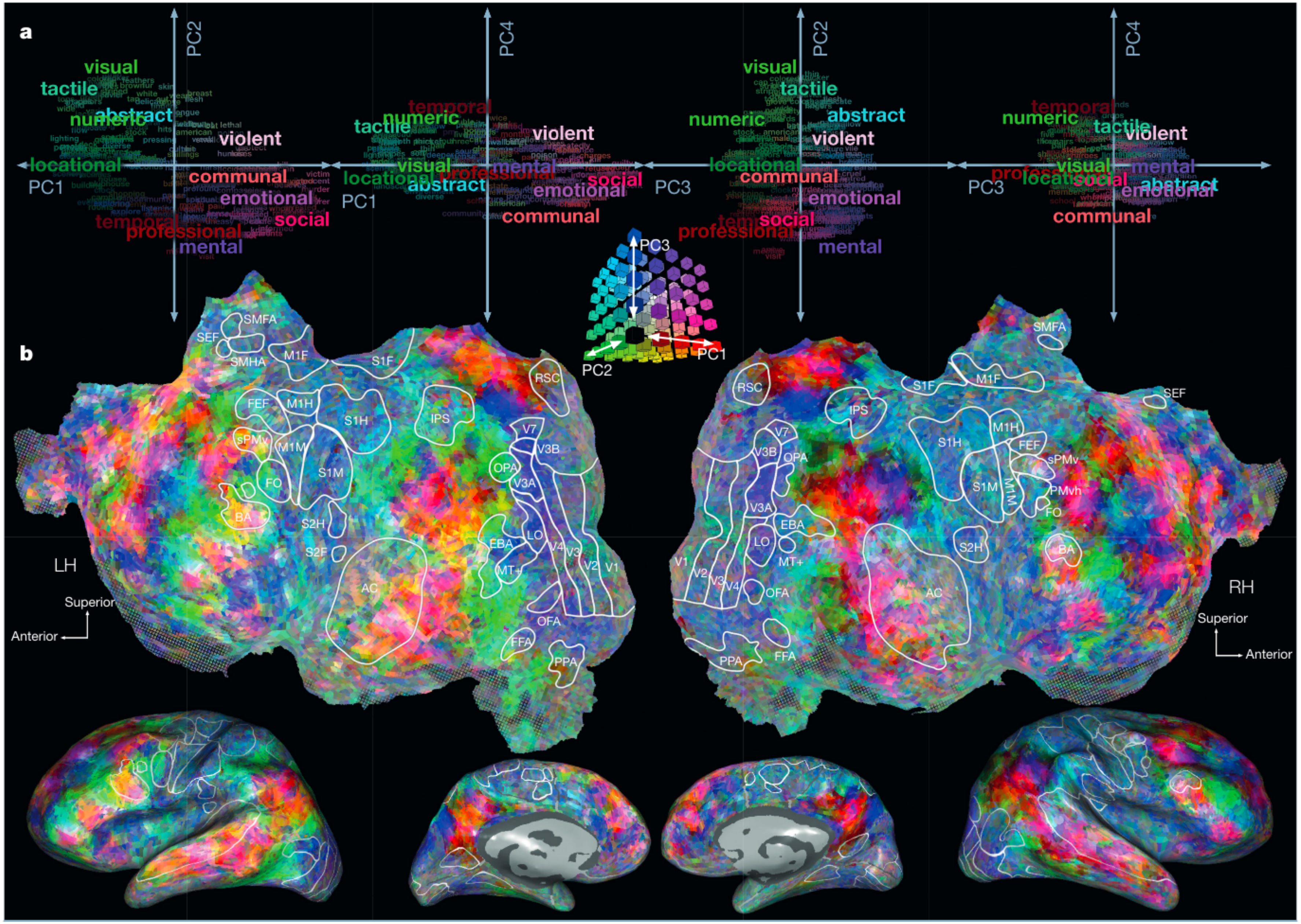
Calculate the correlation between the cosines of the word embeddings and the simlex-999 values

Against human brain data?

Huth et al (2016)

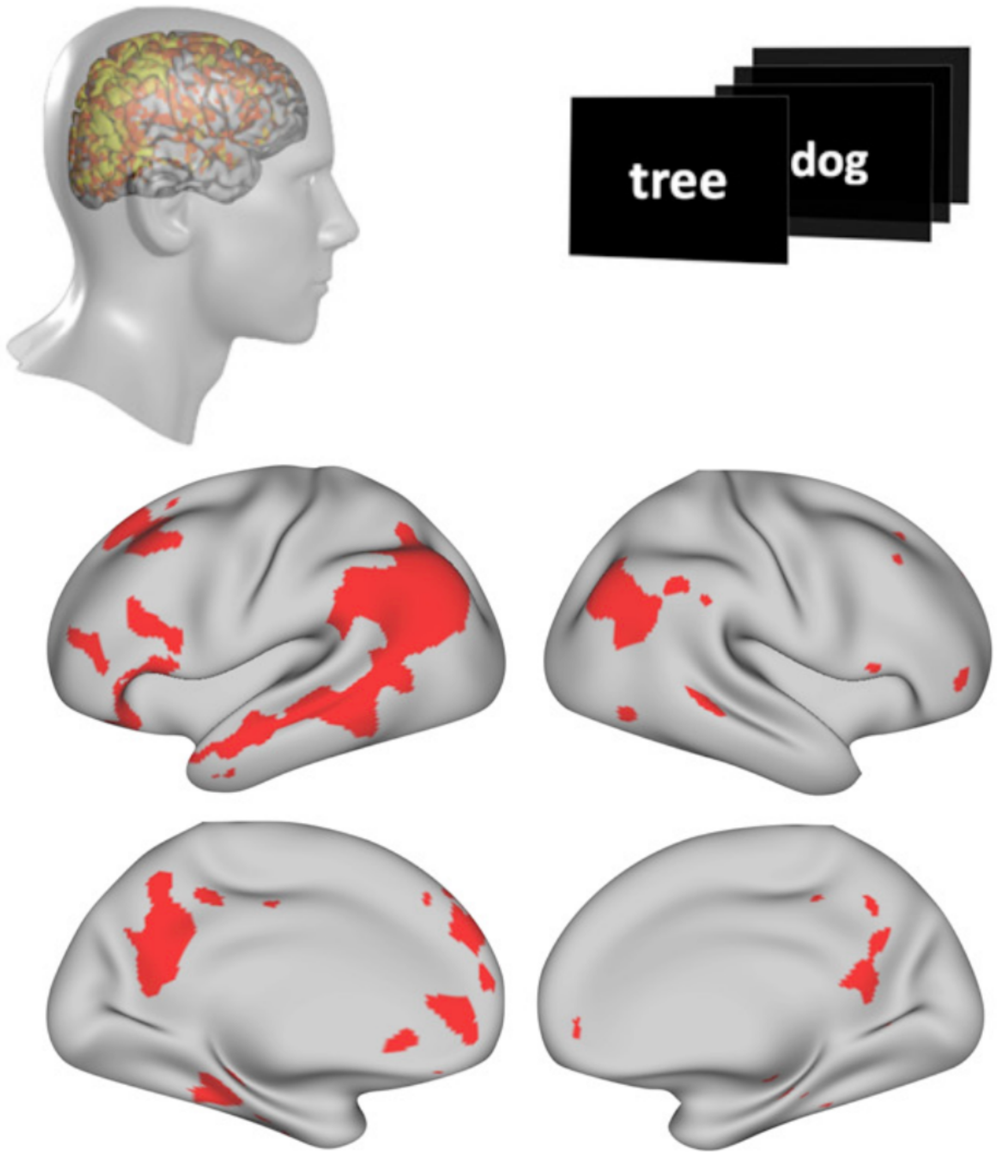


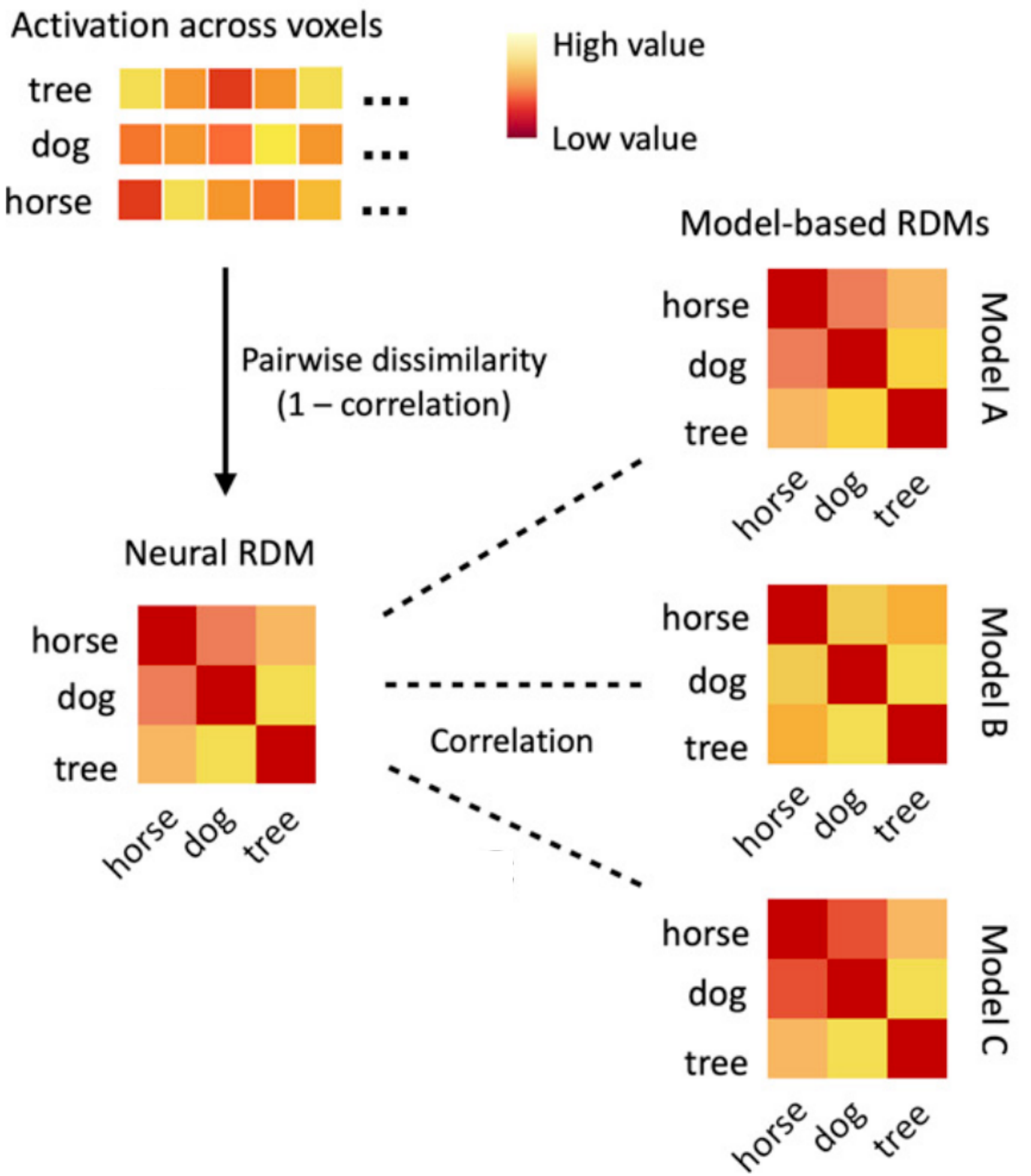
@Jixing Li


a
b
PC1
PC2
PC3
PC4
visual
tactile
abstract
numeric
violent
locational
communal
emotional
social
temporal
professional
mental
LH
RH
Superior
Anterior
SMFA
SEF
SMHA
M1F
S1F
FEF
M1H
S1H
IPS
RSC
sPMv
M1M
S1M
FO
BA
S2H
S2F
AC
V7
V3B
OPA
V3A
LO
EBA
MT+
V4
V3
V2
V1
OFA
FFA
PPA
PMvh

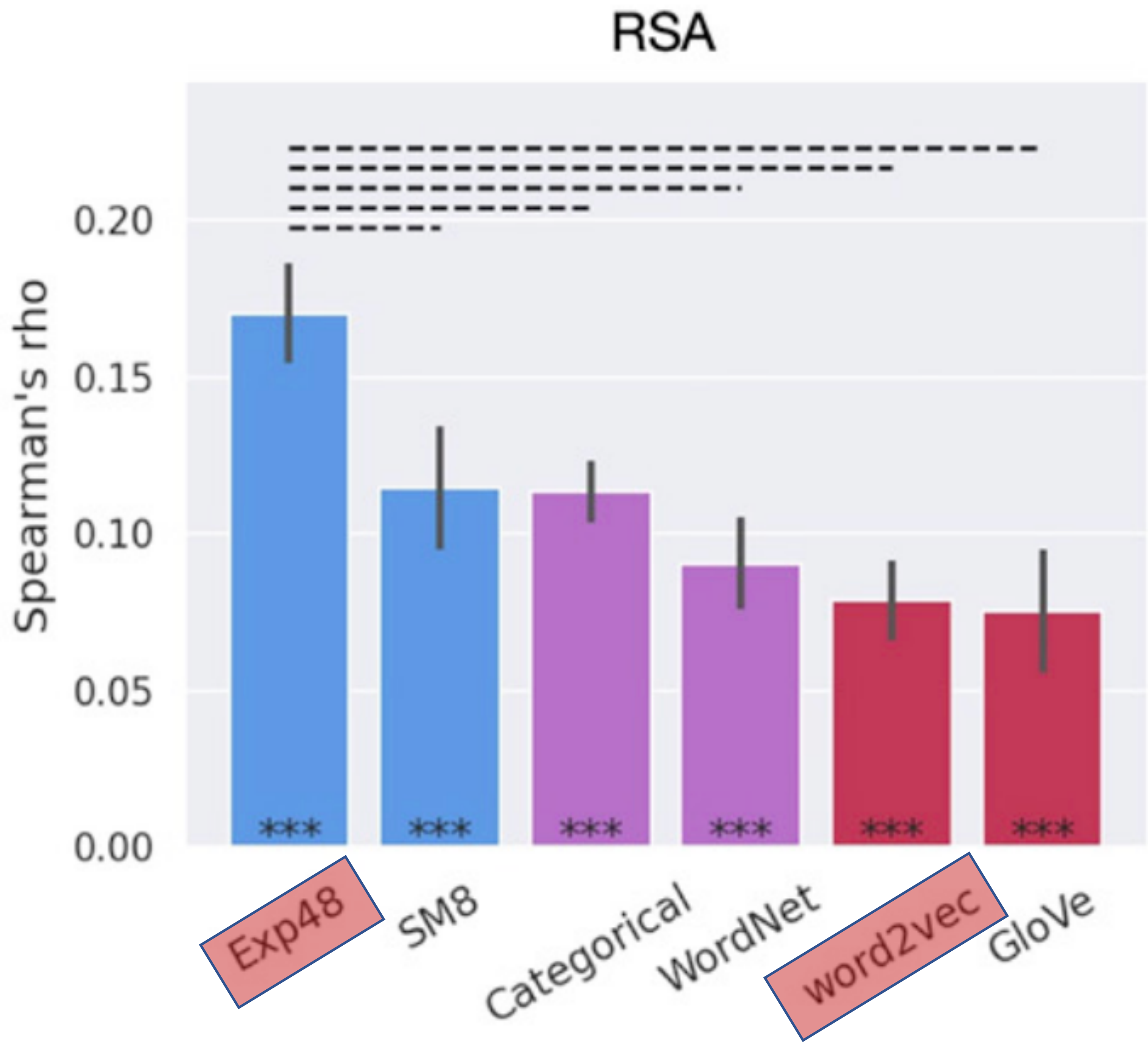
@Jixing Li

Against human brain data?





Against human brain data?



Word	Vision	Bright	Dark	Color	Pattern	Large	Small
ant	3.5484	0.3548	3.5806	3.9355	1.9355	0.0968	5.871
bicycle	5.3	1.1667	0.6333	1	2.1667	1.7	1.2667
farm	5.7097	1.1935	0.5161	1.7419	1.8065	5.0645	0.129
farmer	4.1786	0.5	0.3214	0.4286	0.6071	1.4286	0.6786
green	4.2963	1.7778	1	5.9259	1.5926	0.1852	0.1111
red	5	3.2857	1.25	6	1.4643	0.1071	0.0357
rocket	5.5	2.9333	0.7333	1.8667	1.9	5.6	0.2333
trust	0.3793	0.1379	0.0345	0.3103	0.2069	0.3103	0.069

To do

Do HW10

Textbooks:
Jurafsky and Martin, *Speech and Language Processing*
https://web.stanford.edu/~jurafsky/slp3/
Bird et al. *Natural Language Processing with Python*
https://www.nltk.org/book/

Next lecture: **File** Ch10