Applications: NLP

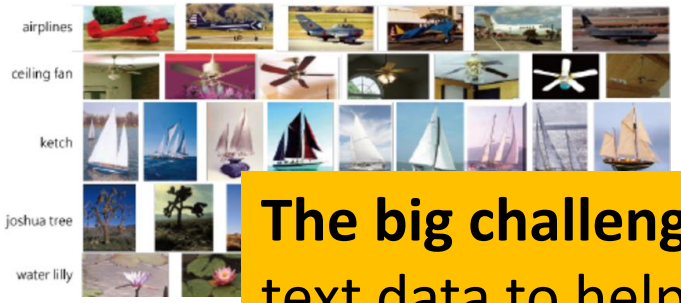
Designing, Visualizing and Understanding Deep Neural Networks

CS W182/282A

Instructor: Sergey Levine
UC Berkeley

The Big Idea: Unsupervised Pretraining

Deep learning works best when we have a lot of data



The big challenge: how can we use **freely available** and **unlabeled** text data to help us apply deep learning methods to NLP?

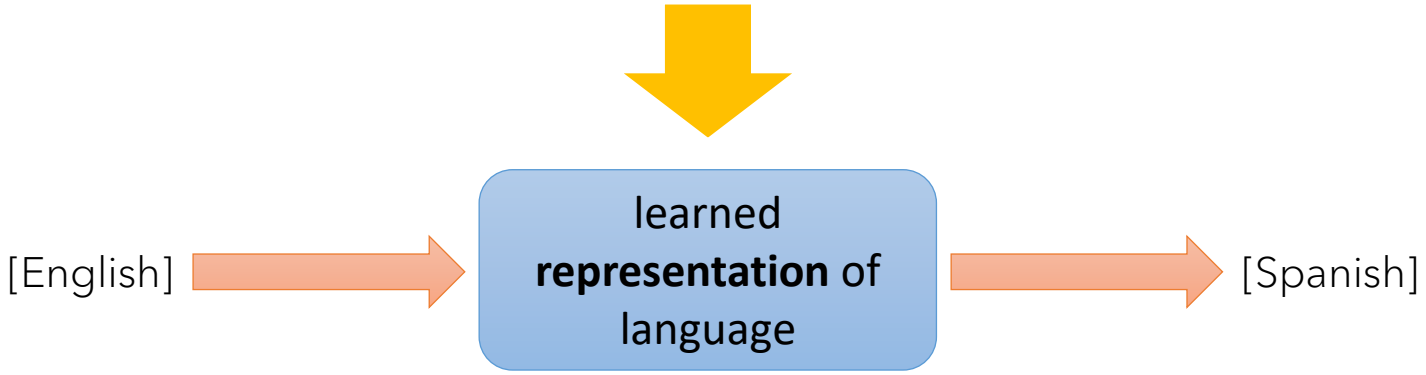
Good news: there is plenty of data of text out there!

Bad news: most of it is unlabeled

1,000s of times more data **without** labels (i.e., valid English text in books, news, web) vs. labeled/paired data (e.g., English/French translations)

What can we do with unlabeled data?





How can we represent these… representations?

local non-contextual representations

global context-dependent representations

word embeddings **sentence embeddings** **pretrained language models**

Start simple: how do we represent **words?**

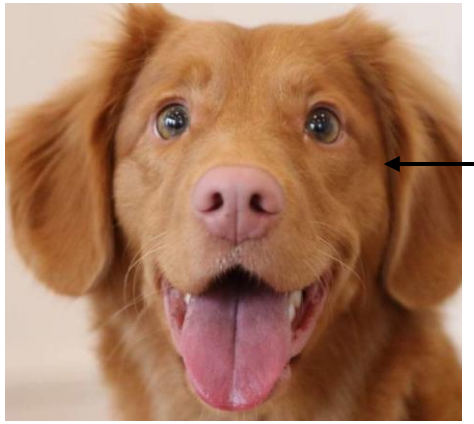
$$x_{1,1} = \begin{bmatrix} 0 \\ 0 \\ . \\ 0 \\ 1 \\ 0 \\ . \\ 0 \end{bmatrix}$$

dimensionality = number of possible words

index of this word

not great, not terrible…

This means basically nothing by itself



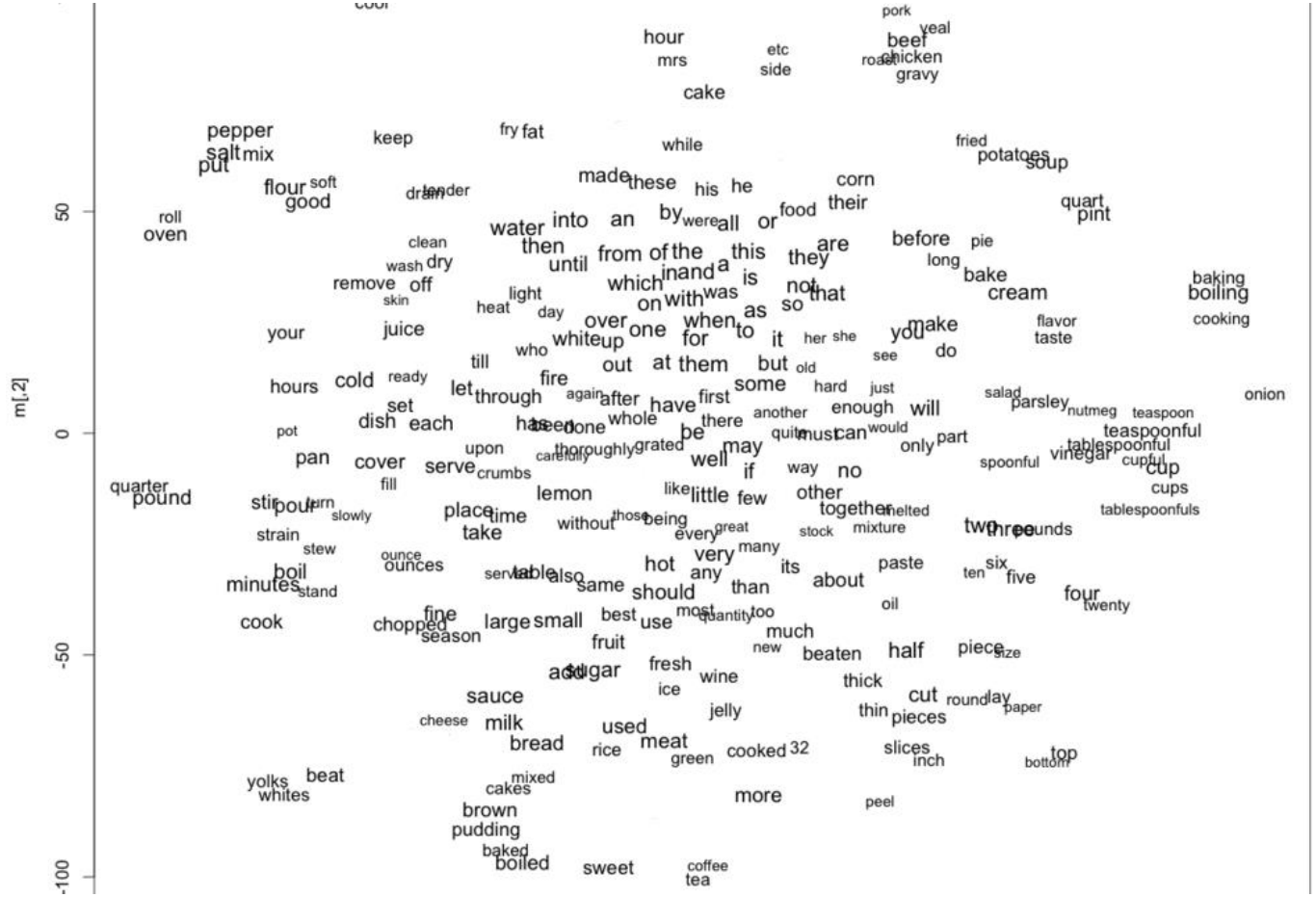
The pixels mean something!

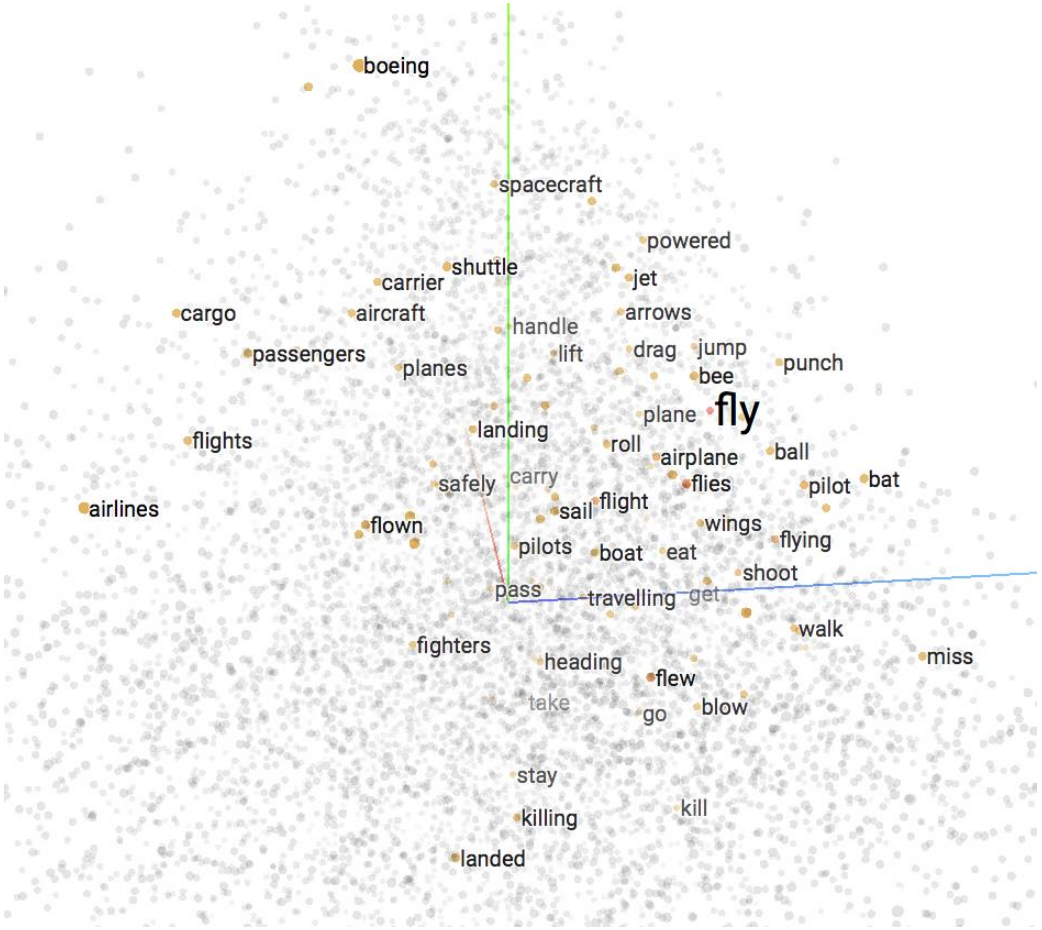
Not much (not a great metric space), but they mean something

Maybe if we had a more meaningful representation of words, then learning downstream tasks would be much easier!

Meaningful = vectors corresponding to **similar** words should be close together

Some examples of **good** embeddings



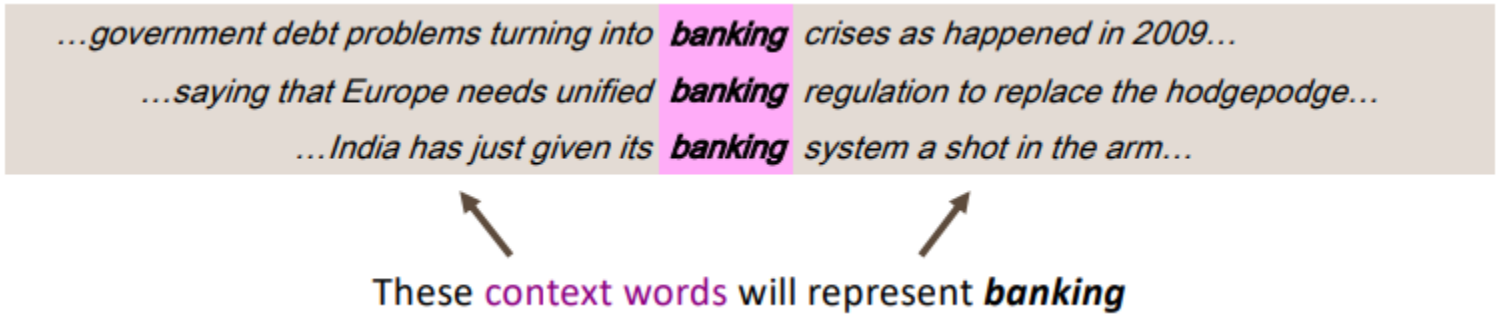


How do we learn embeddings?

Basic idea: the meaning of a word is determined by what **other** words occur in close proximity to it

Put another way: the more interchangeable words are, the more similar they are

Example: Seattle hotel vs. Seattle motel

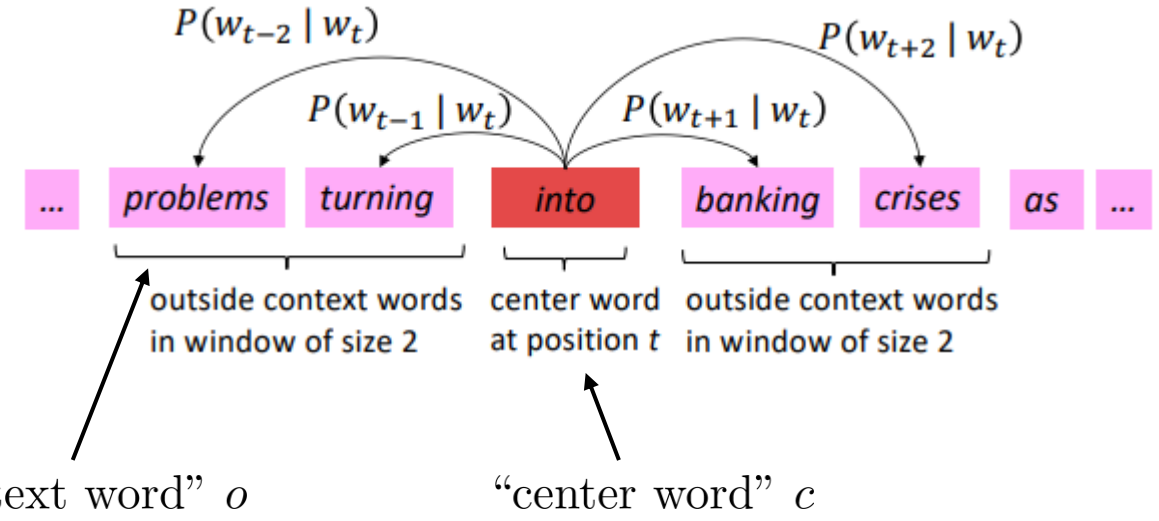


Basic principle: pick a representation for each word such that its neighbors are "close" under this representation

Examples & Images: Christopher Manning, CS224n

More formally…

Can we **predict** the neighbors of a word from its **embedding value**?



(learned) vector representation of o

(learned) vector representation of c

$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

all possible words (vocabulary)

looks a bit like a **logistic regression** model

how to train? $\arg \max_{u_1,\ldots,u_n,v_1,\ldots,v_n} \sum_{c,o} \log p(o|c)$

u and v vectors for all possible words

all possible c and o combinations
e.g., for each word c, pick all words o that are within 5 step

Examples & Images: Christopher Manning, CS224n

word2vec

$$\arg \max_{u_1,\ldots,u_n,v_1,\ldots,v_n} \sum_{c,o} \log p(o|c) \qquad p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

$$\theta = \begin{bmatrix} v_{\text{aardvark}} \\ v_{\text{a}} \\ \ldots \\ v_{\text{zebra}} \\ u_{\text{aardvark}} \\ u_{\text{a}} \\ \ldots \\ u_{\text{zebra}} \end{bmatrix}$$

- Why two vectors? Makes optimization easier
- What to do at the end? Average them
- This then gives us a representation of words!

Mikolov et al. “Linguistic Regularities in Continuous Space Word Representations.” 2013.

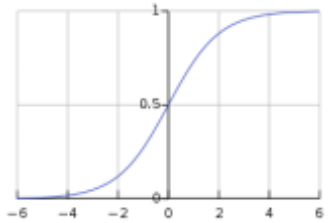
Making word2vec tractable

$$\arg \max_{u_1,\ldots,u_n,v_1,\ldots,v_n} \sum_{c,o} \log p(o|c) \qquad p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Problem: the vocabulary might be **huge**

denominator might be **really costly** to compute

Another idea: what if we instead have a **binary** classification problem ("is this the right word or not")?



$$p(o \text{ is the right word}|c) = \sigma(u_o^T v_c) = \frac{1}{1 + \exp(-u_o^T v_c)}$$

This is not enough! Why?

$$p(o \text{ is the wrong word}|c) = \sigma(-u_o^T v_c) = \frac{1}{1 + \exp(u_o^T v_c)}$$

$$\arg \max_{u_1,\ldots,u_n,v_1,\ldots,v_n} \sum_{c,o} \left(\log p(o \text{ is right}|c) + \sum_{w} \log p(w \text{ is wrong}|c) \right)$$

randomly chosen "negatives"

Mikolov et al. "Linguistic Regularities in Continuous Space Word Representations." 2013.

Making word2vec tractable: summary

$$p(o \text{ is the right word}|c) = \sigma(u_o^T v_c) = \frac{1}{1 + \exp(-u_o^T v_c)}$$

$$p(o \text{ is the wrong word}|c) = \sigma(-u_o^T v_c) = \frac{1}{1 + \exp(u_o^T v_c)}$$

$$\arg \max_{u_1,\ldots,u_n,v_1,\ldots,v_n} \sum_{c,o} \left(\log p(o \text{ is right}|c) + \sum_w \log p(w \text{ is wrong}|c) \right)$$

$$\arg \max_{u_1,\ldots,u_n,v_1,\ldots,v_n} \sum_{c,o} \left(\log \sigma(u_o^T v_c) + \sum_w \log \sigma(-u_w^T v_c) \right)$$

Intuition: push v_c *toward* u_o and *away* from other vectors u_w

Mikolov et al. “Linguistic Regularities in Continuous Space Word Representations.” 2013.

word2vec examples

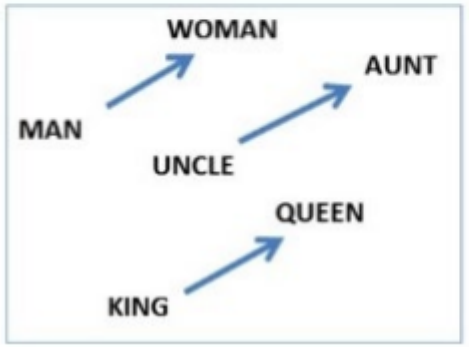
$$\arg \max_{u_1,\ldots,u_n,v_1,\ldots,v_n} \sum_{c,o} \log p(o|c) \qquad p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

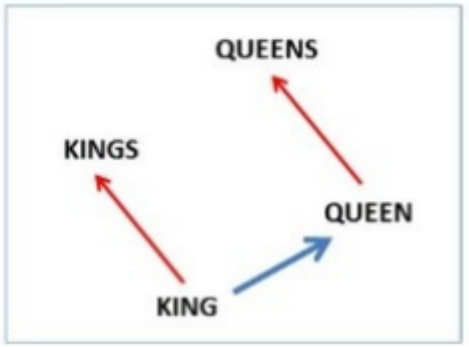
Algebraic relations:

vec(“woman")−vec(“man") ≃ vec(“aunt")−vec(“uncle")

vec(“woman")−vec(“man") ≃ vec(“queen")−vec(“king")

This is a little bit idealized, most relationships are not nearly this “nice”





Mikolov et al. “Linguistic Regularities in Continuous Space Word Representations.” 2013.

word2vec examples

$$\arg \max_{u_1,\ldots,u_n,v_1,\ldots,v_n} \sum_{c,o} \log p(o|c) \qquad p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Word2vec model computed from 6 billion word corpus of news articles

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Mikolov et al. “Linguistic Regularities in Continuous Space Word Representations.” 2013.

word2vec examples

$$\arg \max_{u_1,\ldots,u_n,v_1,\ldots,v_n} \sum_{c,o} \log p(o|c) \qquad p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Mikolov et al. “Linguistic Regularities in Continuous Space Word Representations.” 2013.

Word2vec summary

$$\arg \max_{u_1,\ldots,u_n,v_1,\ldots,v_n} \sum_{c,o} \log p(o|c) \qquad p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

What do we do we with this?

- Use as word representation in place of one-hot vectors
- Much more meaningful for downstream applications
- Can train word embeddings on large unlabeled corpus, and then use it as input into a supervised model trained on a much smaller corpus
- Could think of it as a simple type of **unsupervised pretraining**

Mikolov et al. “Linguistic Regularities in Continuous Space Word Representations.” 2013.