Word Embeddings

CS 6120 Natural Language Processing

Si Wu

Some slides borrowed from Jurafsky & Martin Chapter 5

Logistics

- The first coding assignment is due this Friday midnight
  - Try to submit on Gradescope as soon as possible if you are not familiar with Gradescope
  - If you need help, reach out to TAs and ask on Ed Discussion
- Today is the last day to drop a class without a W
- I've posted links to materials for reviewing machine learning basics on Ed Discussion.

- Today: word embedding and its applications
  - You probably already learned the math from vector calculus, but we will focus more on the fun applications and think more deeply about language

Feature vector using word count

- Last time we talked about if the vector length is |V| where V is the set of vocabulary, the vector will be too **sparse** since most of the entries will be 0
- Today we will introduce simple word embeddings that are **dense** and perform better at many NLP tasks

What's a word embedding?

Polysemy: having multiple meanings or word senses

- In this lecture, we are talking about the **static word embeddings**
  - The vector representation of word that are learned directly from the distribution of text
  - "Apple is a company" and "I just ate an apple", the word apple will have the same embedding (if these two sentences are in the same training data).
    - Word order and sense don't matter, even if the word has multiple meanings
- In the future, once we learn about neural networks and and other advanced LMs, we will introduce **contextualized embeddings** for words and sentences

Similarity in NLP

- “This album is awesome”
- “This album is phenomenal”
- “This album is the GOAT”
- “This album is sick”

All these sentences express the same sentiment towards an album but just are said in different ways

Similarity in NLP

But also similarity between sentences:

paraphrase of the same language, paraphrase of different languages, aka. translations!

- What can be similar between two words?
  - The environment / context they are in
    - Travel: flight, delay, ticket, airport, suitcase, carry-on, luggage
  - The meaning (synonymy)
    - Happy, joyful, cheerful (the definition of same meaning is loose here)
  - The connotation/sentiment
    - See the meme on this page
  - The emotion
    - Horror: horrifying, terrifying, horrific, scary
  - The language id:
    - French: bonjour, merci, salut, bonsoir, oui,
  - The topic
    - Sports: Celtics, Lakers, Basketball, Boston, Los Angeles
  - The time period of usage:
    - Old English: thou, thee, ye, thy, wilt
  - Etc., you can come up with one for your project!

laptop

classroom

university

study

college

etudiant (French)

Student

学生 (Chinese)

professor

pupil

Words that are **related/associated** with “student”

classroom

laptop

university

study

college

Word association task

Student

professor

Words that are **similar/synonyms** to “student”

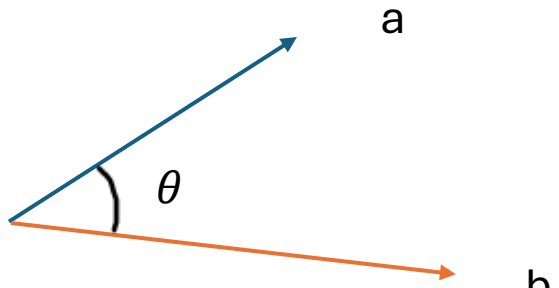
etudiant (French)

Student

学生 (Chinese)

pupil

Dot product review



Dot product of 2 vectors

- Coordinate definition

$$\mathbf{a} = [a_1, a_2, \cdots, a_n]$$

$$\mathbf{b} = [b_1, b_2, \cdots, b_n]$$

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

Dot product can also be expressed in terms of the angle between the two vector

- Geometric definition

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\theta$$

Cosine similarity

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \, \|\mathbf{B}\| \cos\theta$$

Range:
-1 to 1

Cosine similarity = $$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Angle θ

- $\theta = 0, \cos(\theta) = 1$, maximum similarity
- $\theta = 90, \cos(\theta) = 0$, unrelated
- $\theta = 180, \cos(\theta) = -1$, opposite direction

Cosine similarity is just the normalized dot product

Cosine similarity vs dot product

Raw dot product:

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

Because the dot product between a and b is the sum of the products of their component in each dimension, the dot product value is high when the vectors have large values in the same dimensions

Cosine similarity vs dot product

Raw dot product:

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

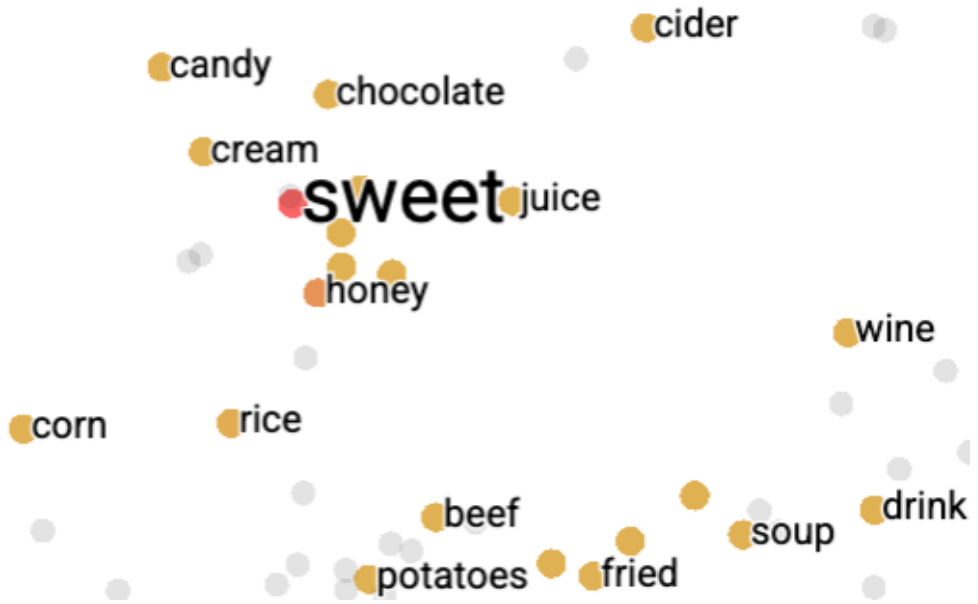
However, this is problematic because frequent words like “the” and “of” have longer vector, and dot product overly favors them

This is why most of the time in NLP we will use cosine similarity of two vectors instead of raw dot product.

Now, instead of vector a, let’s think about a word embedding

Word association and context

- One of the many NLP tasks that uses “similarity”
- Here, we think about the topic and the environment/surroundings that a word belongs to
- Some words are often seen together, sharing the same surroundings
  - They are in the same “neighborhood”
  - We want to find these closely related words
- But first, we need to embed them into a vector space, before we can find these clusters



Getting the embeddings

- We decide the size of the dimension that we will be computing on
- Using the distributional information of a text corpus, we map these words onto the vector space, then we can discover patterns based on their relative distances to each other
- So how to get these word embeddings?
  - There are many methods, word2vec (Mikolov et al.), GloVe (Pennington et al.)
  - We will talk about word2vec in this lecture

Word2vec and skip-gram

Intuition: Consider these examples

“I want to add some ____ to my coffee”

sugar, milk, cream

These words are not similar but related to coffee

“___ is sweet”

sugar, candy, lollipop

These words are (almost) similar in meaning

Skip-gram with negative sampling (SGNS)

- Word2vec method has many options (different objective functions),
  - one of them is skip-gram with negative sampling (SGNS)
- Intuition: the natural co-occurrence of words can be turned into data that we train on
  - if they do co-occur within a window length → positive example
  - Randomly select other words as negative example
  - No need to label any data.
  - This is called **self-supervision:** using the data itself to generate labels/ supervisory signals

Approach: predict if candidate word *c* is a "neighbor"

1. Treat the target word *t* and a neighboring context word *c* as **positive examples**.
2. Randomly sample other words in the lexicon to get negative examples
3. Use logistic regression to train a classifier to distinguish those two cases
4. Use the learned weights as the embeddings

Training sentence:

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

c1 c2 w c3 c4

Assume a +/- 2 word window

Goal: train a classifier that is given a candidate (**w**ord, **c**ontext) pair

+ positive example: (apricot, jam)

- negative example: (apricot, mitochondria)

…

And assigns each pair a probability:

$P(+|w, c)$

$P(-|w, c) = 1 - P(+|w, c)$

Training sentence:

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

c1 c2 w c3 c4

The intuition is that:

if a word's embedding will be **nearby** another word's embedding if these two words are **similar**

- Mathematically, if two embeddings are nearby, their dot product will be high
- Since cosine similarity is a normalized dot product
- We assume that their cosine similarity will be high as well.

Turning dot products into probabilities

- w is the target word vector, c is one of target word's context word's vector
- Cosine similarity(w,c) $\propto w \cdot c$

To turn this into a probability

- We'll use the sigmoid from logistic regression:

$$P(+|w,c) = \sigma(c \cdot w) = \frac{1}{1+\exp(-c \cdot w)}$$

$$P(-|w,c) = 1 - P(+|w,c)$$

$$= \sigma(-c \cdot w) = \frac{1}{1+\exp(c \cdot w)}$$

How Skip-Gram Classifier computes $P(+|w, c)$

$$P(+|w,c) = \sigma(c \cdot w) = \frac{1}{1+\exp(-c \cdot w)}$$

This is for one context word, but we have lots of context words.
We'll assume independence and just multiply them:

$$P(+|w,c_{1:L}) = \prod_{i=1}^{L} \sigma(c_i \cdot w)$$

$$\log P(+|w,c_{1:L}) = \sum_{i=1}^{L} \log \sigma(c_i \cdot w)$$

Log for easier math

product → sum

Skip-gram classifier: summary

A probabilistic classifier, given

- a test target word *w*
- its context window of *L* words $c_{1:L}$

Estimates probability that w occurs in this window based on similarity of w (embeddings) to $c_{1:L}$ (embeddings).

To compute this, we just need embeddings for all the words.

Skip-Gram Training data

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

c1 c2 w 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

Word2vec: how to learn vectors

- Given the set of positive and negative training instances, and an initial set of embedding vectors
- The goal of learning is to adjust those word vectors such that we:
  - **Maximize** the similarity of the target word, context word pairs (w , c_{pos}) drawn from the positive data
  - **Minimize** the similarity of the (w , c_{neg}) pairs drawn from the negative data.

Loss function for one *w with* c_{pos} , c_{neg1} …c_{negk}

Maximize the probability/similarity of the target with the actual context words.

Maximize the probability of neg samples being non-neighbors → Minimize the similarity of the target with the *k* negative sampled non-neighbor words.

$$L_{CE} = -\log\left[P(+|w,c_{pos})\prod_{i=1}^{k}P(-|w,c_{neg_i})\right]$$

Multiply because we assume independence

$$= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log P(-|w,c_{neg_i})\right]$$

$$= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log\left(1-P(+|w,c_{neg_i})\right)\right]$$

$$= -\left[\log \sigma(c_{pos}\cdot w) + \sum_{i=1}^{k}\log\sigma(-c_{neg_i}\cdot w)\right]$$

$$P(+|w,c) = \sigma(c\cdot w) = \frac{1}{1+\exp(-c\cdot w)}$$

Learning the classifier

- How to learn?
  - Stochastic 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.

Reminder: gradient descent

- At each step
  - Direction: We move in the reverse direction from the gradient of the loss function to minimize loss
  - Magnitude: we move the value of this gradient $\frac{d}{dw} L(f(x; w), y)$ weighted by a **learning rate** η
  - Higher learning rate means move *w* faster

$$w^{t+1} = w^{t} - h \frac{d}{dw} L(f(x; w), y)$$

The derivatives of the loss function

$$L_{CE} = -\left[\log \sigma(c_{pos} \cdot w) + \sum_{i=1}^{k} \log \sigma(-c_{neg_i} \cdot w)\right]$$

To get gradient, we need to take the derivatives with respect to pos and neg

$$\frac{\partial L_{CE}}{\partial c_{pos}} = [\sigma(c_{pos} \cdot w) - 1]w$$

$$\frac{\partial L_{CE}}{\partial c_{neg}} = [\sigma(c_{neg} \cdot w)]w$$

$$\frac{\partial L_{CE}}{\partial w} = [\sigma(c_{pos} \cdot w) - 1]c_{pos} + \sum_{i=1}^{k} [\sigma(c_{\mathrm{neg}_i} \cdot w)]c_{neg_i}$$

Update equation in SGD

$$w^{t+1} = w^t - h \frac{d}{dw} L(f(x; w), y)$$

Start with randomly initialized C and W matrices, then incrementally do updates

$$c_{pos}^{t+1} = c_{pos}^t - \eta [\sigma(c_{pos}^t \cdot w^t) - 1] w^t$$

Plug in the derivatives

$$c_{neg}^{t+1} = c_{neg}^t - \eta [\sigma(c_{neg}^t \cdot w^t)] w^t$$

$$w^{t+1} = w^t - \eta \left[[\sigma(c_{pos} \cdot w^t) - 1] c_{pos} + \sum_{i=1}^{k} [\sigma(c_{neg_i} \cdot w^t)] c_{neg_i} \right]$$

Two sets of embeddings

SGNS learns two sets of embeddings

Target embeddings matrix W

Context embedding matrix C

It's common to just add them together, representing word *i* as the vector $w_i + c_i$

Summary: How to learn word2vec (skip-gram) embeddings

Start with V random d-dimensional vectors as initial embeddings

Train a classifier based on embedding similarity

- Take a corpus and take pairs of words that co-occur as positive examples
- Take pairs of words that don't co-occur as negative examples
- Train the classifier to distinguish these by slowly adjusting all the embeddings to improve the classifier performance
- Throw away the classifier code and keep the embeddings.

The kinds of neighbors depend on window size

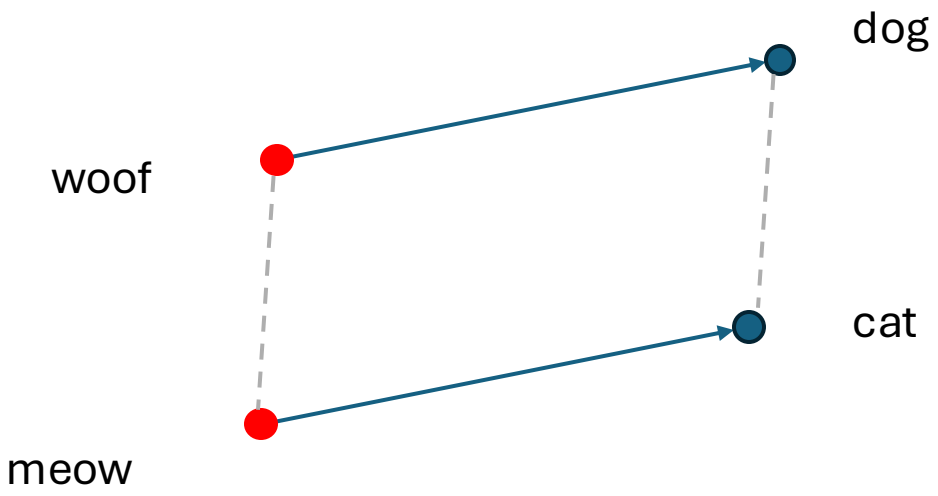
Window size is a parameter we can tune on dev/val set, and it affects performance.

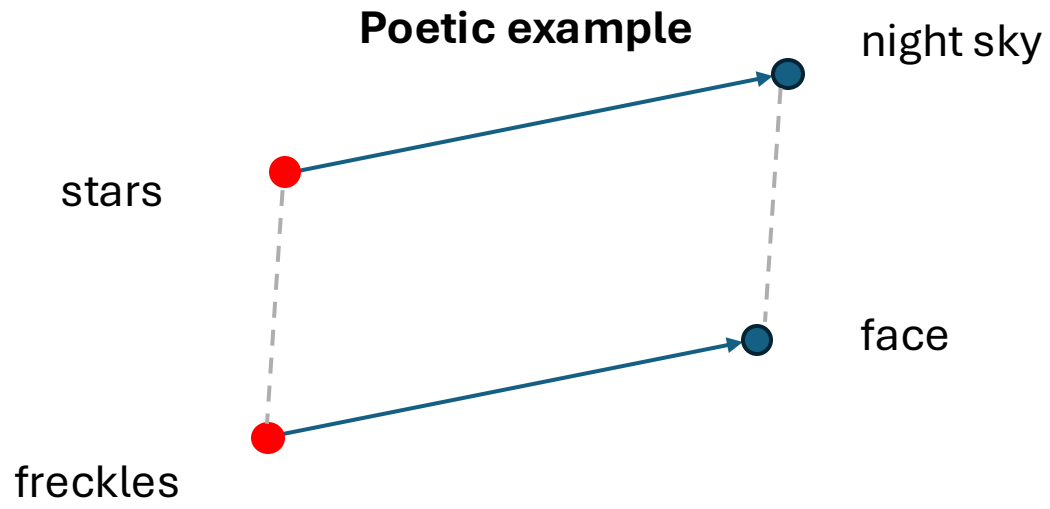
- **Small windows** (C= +/- 2) : nearest words are syntactically similar words in same taxonomy
  - *Hogwarts* nearest neighbors are other fictional schools
    - *Sunnydale, Evernight, Blandings*
- **Large windows** (C= +/- 5) : nearest words are related words in same semantic field
  - *Hogwarts* nearest neighbors are Harry Potter world:
    - *Dumbledore, half-blood, Malfoy*

Analogy

A to B (is like)/≈ X to what?

- E.g. woof to dog is like meow to ?

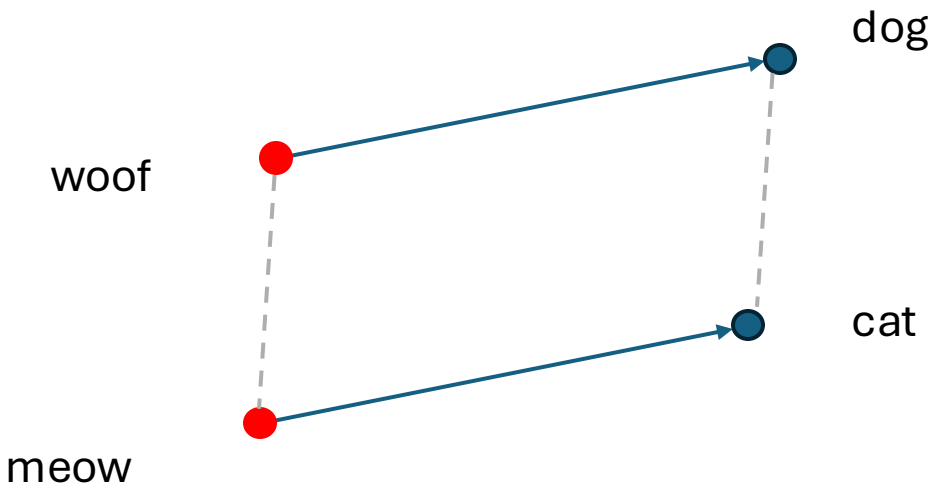




Analogy

Rumelhart and Abrahamson (1973) proposed the parallelogram model

- A paper from cognitive psychology
- Survey on lower-division psychology class students from UCSD



Example: apple:tree::grape: _____

A. bush
B. vine
C. barrel
D. ground

1. B

2. D
3. A
4. C

Caveats with the parallelogram method

- It only seems to work for frequent words, small distances and certain relations (relating countries to capitals, or parts of speech), but not others. (Linzen 2016, Gladkova et al. 2016, Ethayarajh et al. 2019a)

- Understanding analogy is an open area of research (Peterson et al. 2020)

Embeddings reflect cultural bias!

Bolukbasi, Tolga, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings." In *NeurIPS*, pp. 4349-4357. 2016.

- Ask “Paris : France :: Tokyo : x”
  - x = Japan
- Ask “father : doctor :: mother : x”
  - x = nurse
- Ask “man : computer programmer :: woman : x”
  - x = homemaker

Algorithms that use embeddings as part of e.g., hiring searches for programmers, might lead to bias in hiring

Gender stereotype *she-he* analogies.

sewing-carpentry	register-nurse-physician	housewife-shopkeeper
nurse-surgeon	interior designer-architect	softball-baseball
blond-burly	feminism-conservatism	cosmetics-pharmaceuticals
giggle-chuckle	vocalist-guitarist	petite-lanky
sassy-snappy	diva-superstar	charming-affable
volleyball-football	cupcakes-pizzas	hairdresser-barber

Gender appropriate *she-he* analogies.

queen-king	sister-brother	mother-father
waitress-waiter	ovarian cancer-prostate cancer	convent-monastery

Bolukbasi, Tolga, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings." In *NeurIPS*, pp. 4349-4357. 2016.

Extreme *she* occupations

1. homemaker	2. nurse	3. receptionist
4. librarian	5. socialite	6. hairdresser
7. nanny	8. bookkeeper	9. stylist
10. housekeeper	11. interior designer	12. guidance counselor

Extreme *he* occupations

1. maestro	2. skipper	3. protege
4. philosopher	5. captain	6. architect
7. financier	8. warrior	9. broadcaster
10. magician	11. figher pilot	12. boss

Bolukbasi, Tolga, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings." In *NeurIPS*, pp. 4349-4357. 2016.

Tracking the history of the meaning/usage of a word

This is projected onto a 2D space using tSNE for visualization

For example, these figures shows a visualization of changes in meaning in English words over the last two centuries, computed by building separate embedding spaces for each decade from historical corpora like Google N-grams and the Corpus of Historical American English.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. Proceedings of ACL.

Historical embedding as a tool to study cultural biases

Garg, N., Schiebinger, L., Jurafsky, D., and Zou, J. (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. Proceedings of the National Academy of Sciences 115(16), E3635–E3644.

- Compute a **gender or ethnic bias** for each adjective: e.g., how much closer the adjective is to "woman" synonyms than "man" synonyms, or names of particular ethnicities
  - Embeddings for **competence** adjective (*smart, wise, brilliant, resourceful, thoughtful, logical)* are biased toward men, a bias slowly decreasing 1960-1990
  - Embeddings for **dehumanizing** adjectives (barbaric, monstrous, bizarre) were biased toward Asians in the 1930s, bias decreasing over the 20th century.
    - These match the results of old surveys done in the 1930s, and the bias decreased in both text and surveys over the 20th century