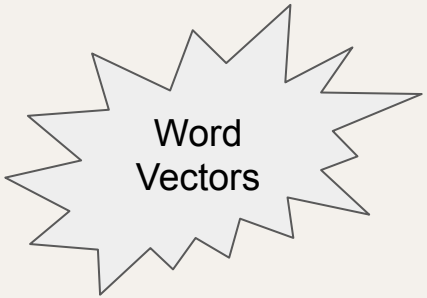
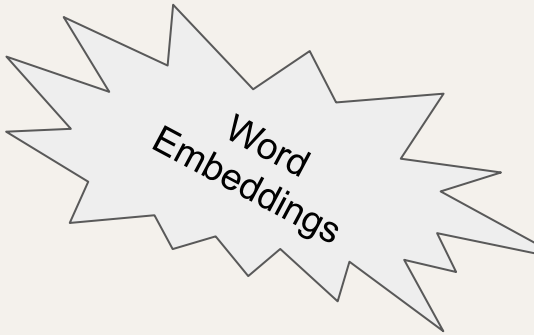
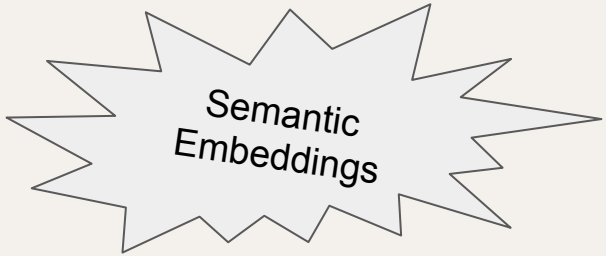
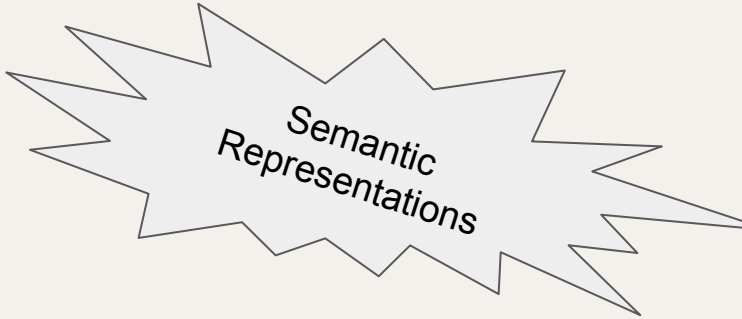




Vector Representations

COMS4054A
COMS7066A
Natural Language Processing





Word2Vec: A Prediction Based Method

Main idea: We need to put information about word contexts into word representation.

How: Learn word vectors by **teaching** them to **predict contexts**.

Word2Vec: A Prediction Based Method

Learned parameters: word vectors

Goal: make each vector “know” about the contexts of its word

How: train vectors to predict possible contexts from words (or, alternatively, words from contexts)

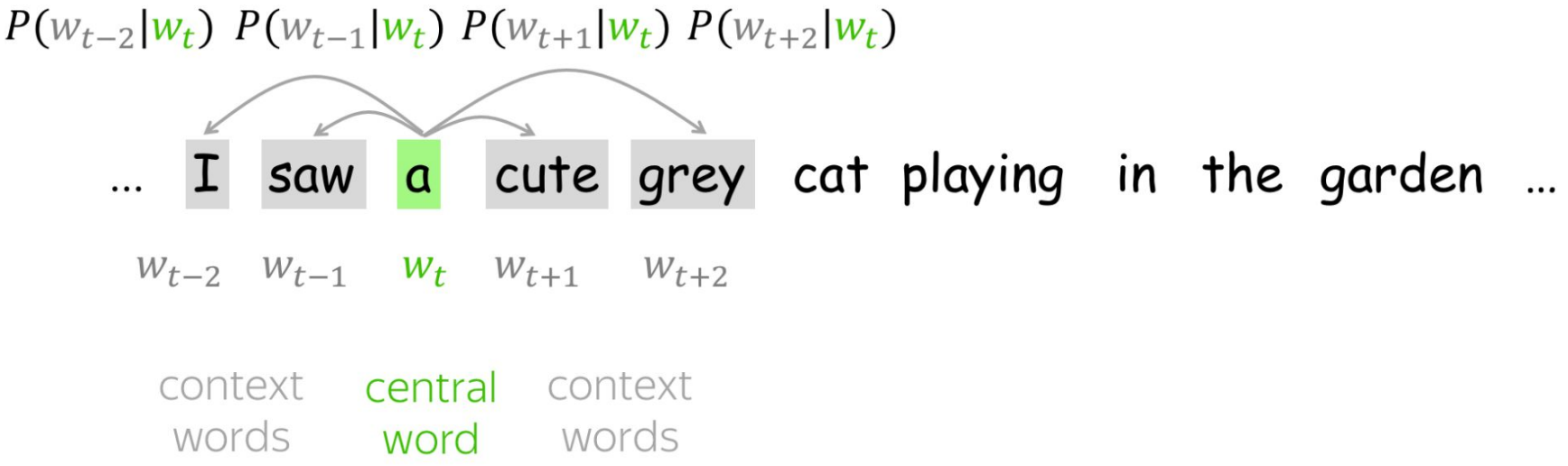
Word2Vec: A Prediction Based Method

An **iterative** method:

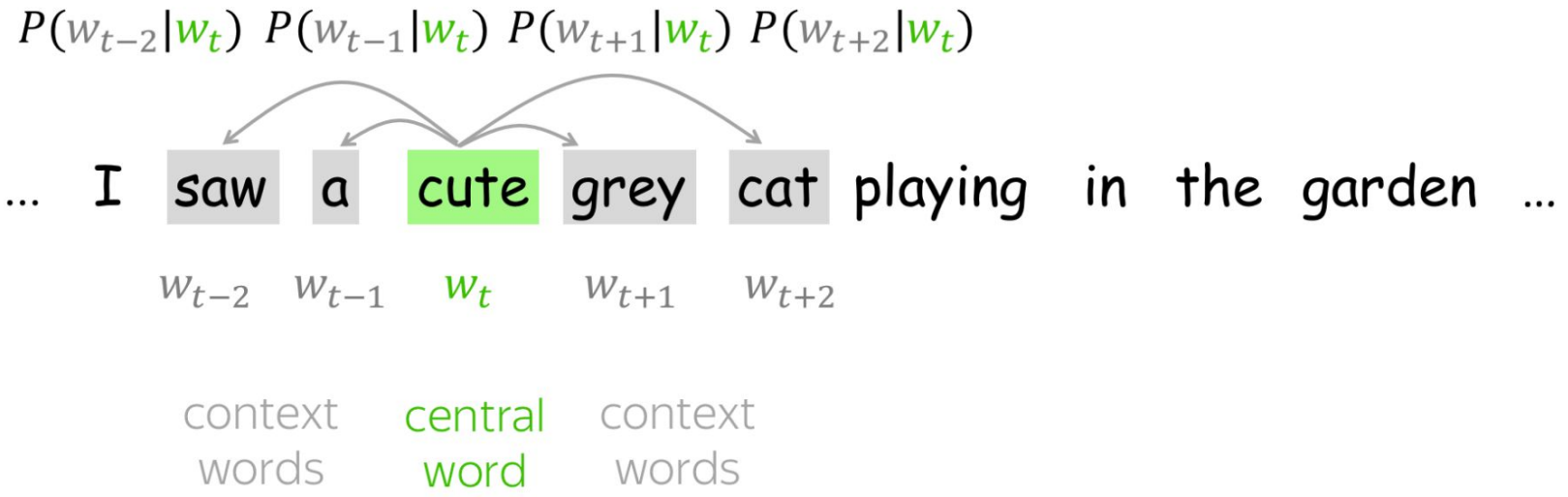
Given a huge text corpus;

1. go over the text with a sliding window, moving one word at a time.
2. At each step, there is a central word and context words (other words in this window);
    - for the central word, compute probabilities of context words;
    - adjust the vectors to increase these probabilities.

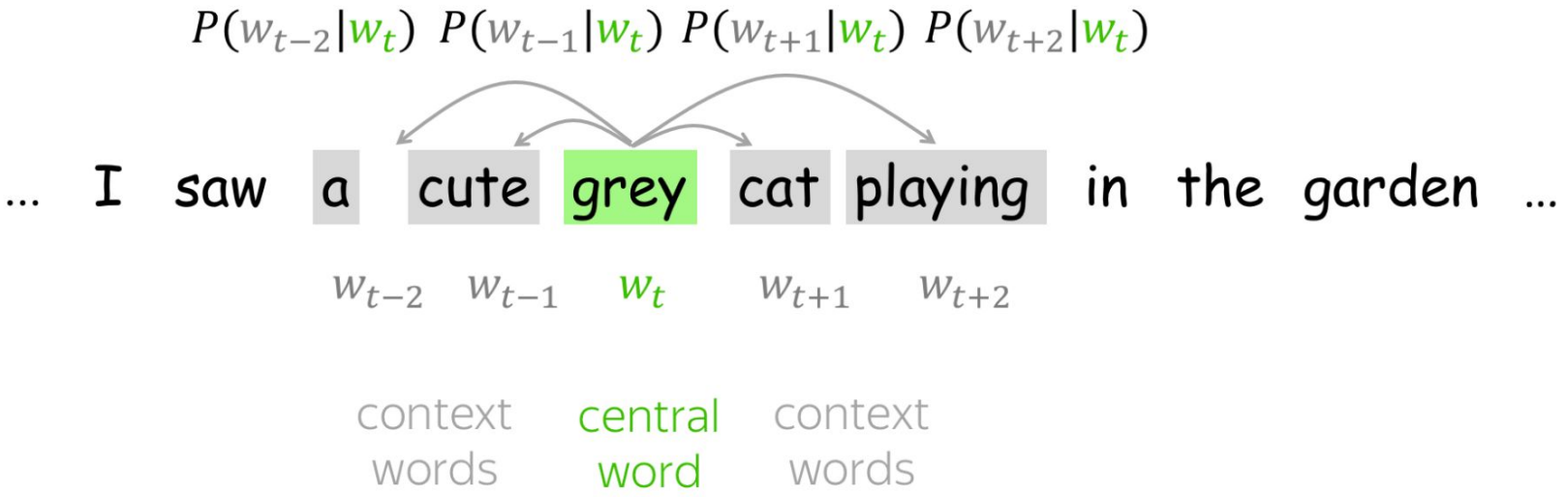
Word2Vec: A Prediction Based Method



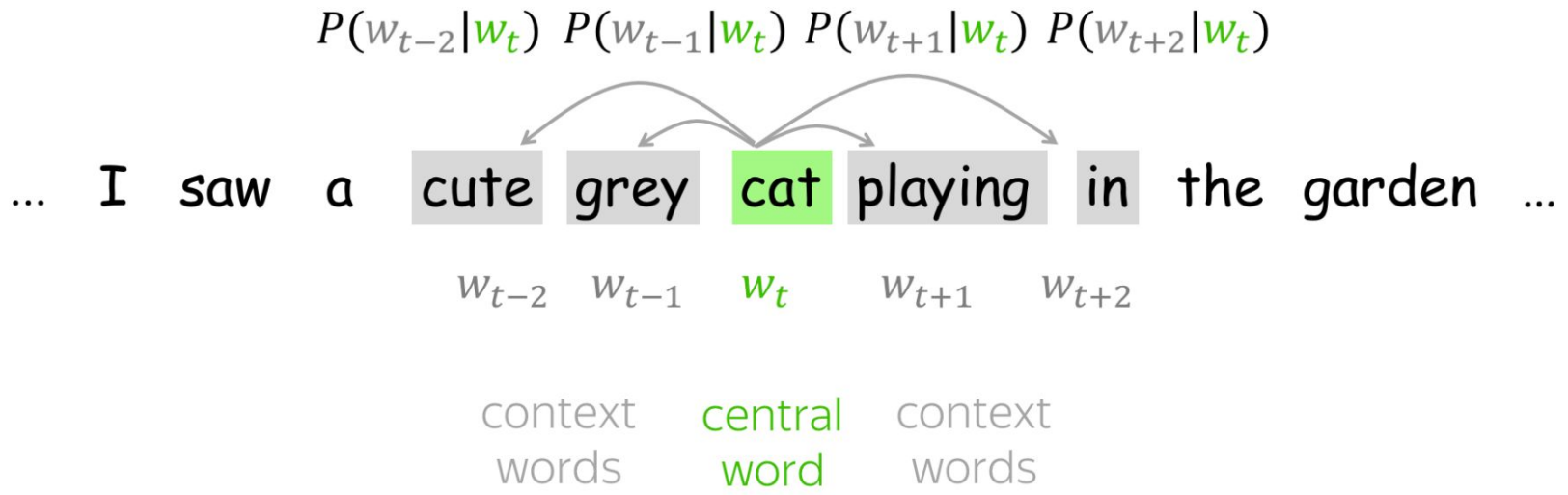
Word2Vec: A Prediction Based Method



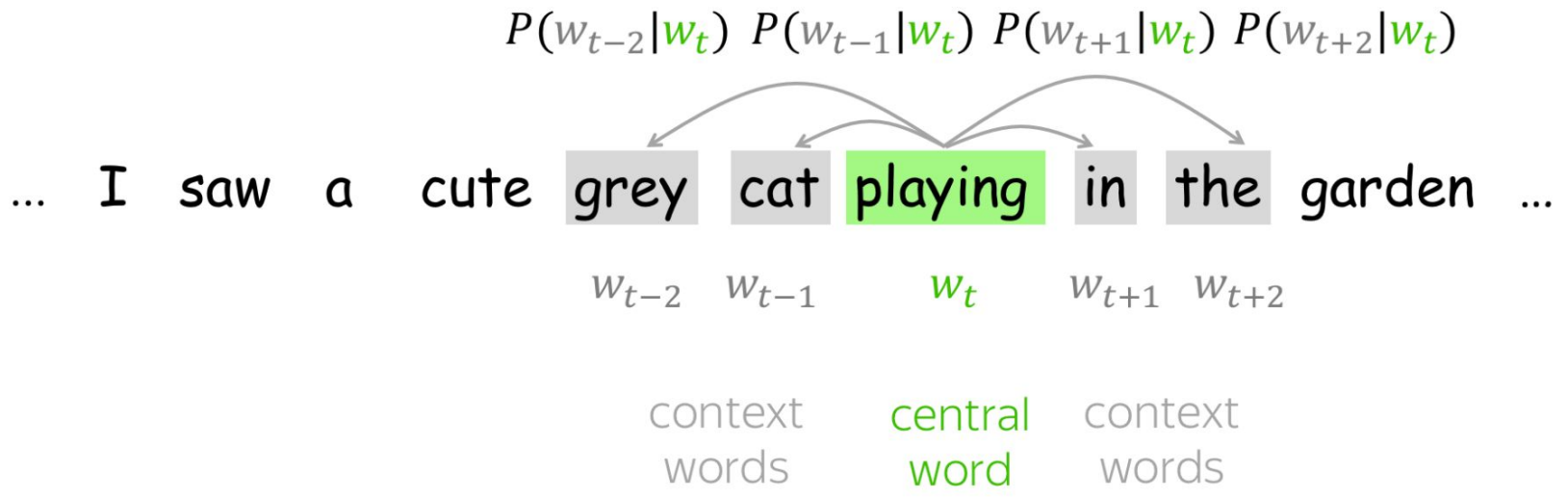
Word2Vec: A Prediction Based Method



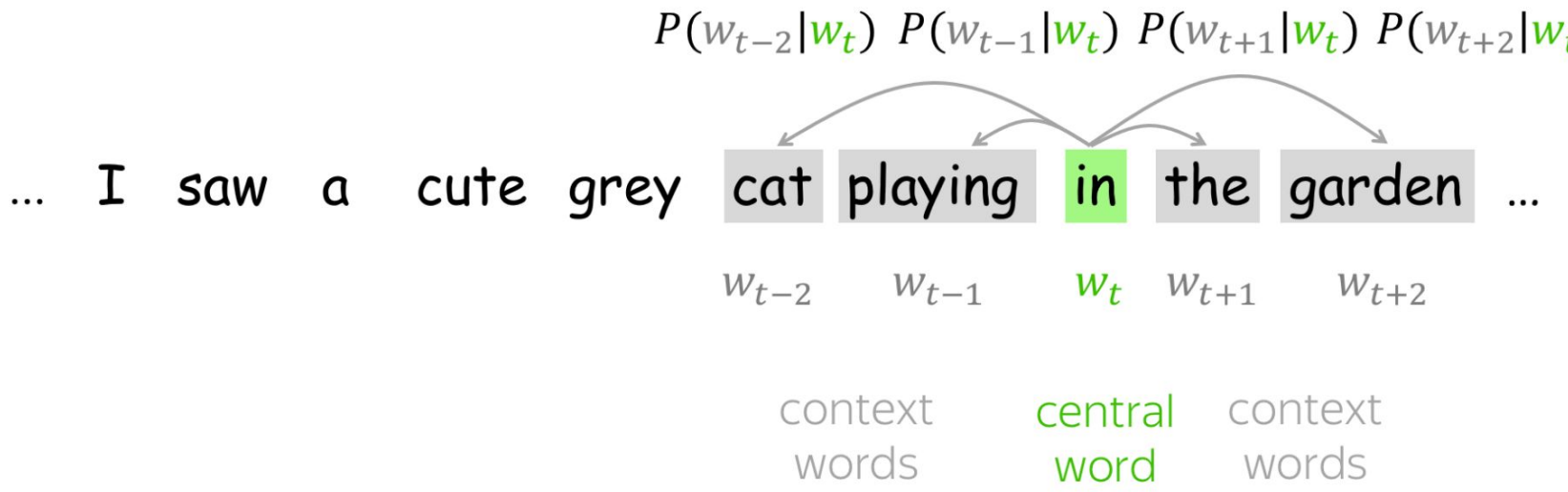
Word2Vec: A Prediction Based Method



Word2Vec: A Prediction Based Method



Word2Vec: A Prediction Based Method



Objective Function: Negative Log-Likelihood

For each position $t = 1, \ldots, T$ in a text corpus, Word2Vec predicts context words within a m-sized window given the central word w_t:

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{-m \le j \le m, j \neq 0} P(w_{t+j} | w_t, \theta),$$

where θ are all variables to be optimized. The objective function (aka loss function or cost function) $J(\theta)$ is the average negative log-likelihood:

$$\text{Loss} = J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m, \\ j \neq 0}} \log P(w_{t+j} | w_t, \theta)$$

agrees with our plan above ⟼ go over text | with a sliding window | compute probability of the context word given the central

Note how well the loss agrees with our plan main above: go over text with a sliding window and compute probabilities. Now let's find out how to compute these probabilities.

Objective Function: Negative Log-Likelihood

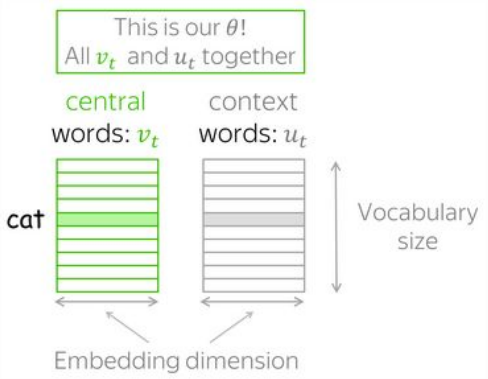
How to calculate $P(w_{t+j}|w_t, \theta)$?

For each word w we will have two vectors:

- v_w when it is a central word;
- u_w when it is a context word.

(Once the vectors are trained, usually we throw away context vectors and use only word vectors.)

Then for the central word c (c - central) and the context word o (o - outside word) probability of the context word is



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

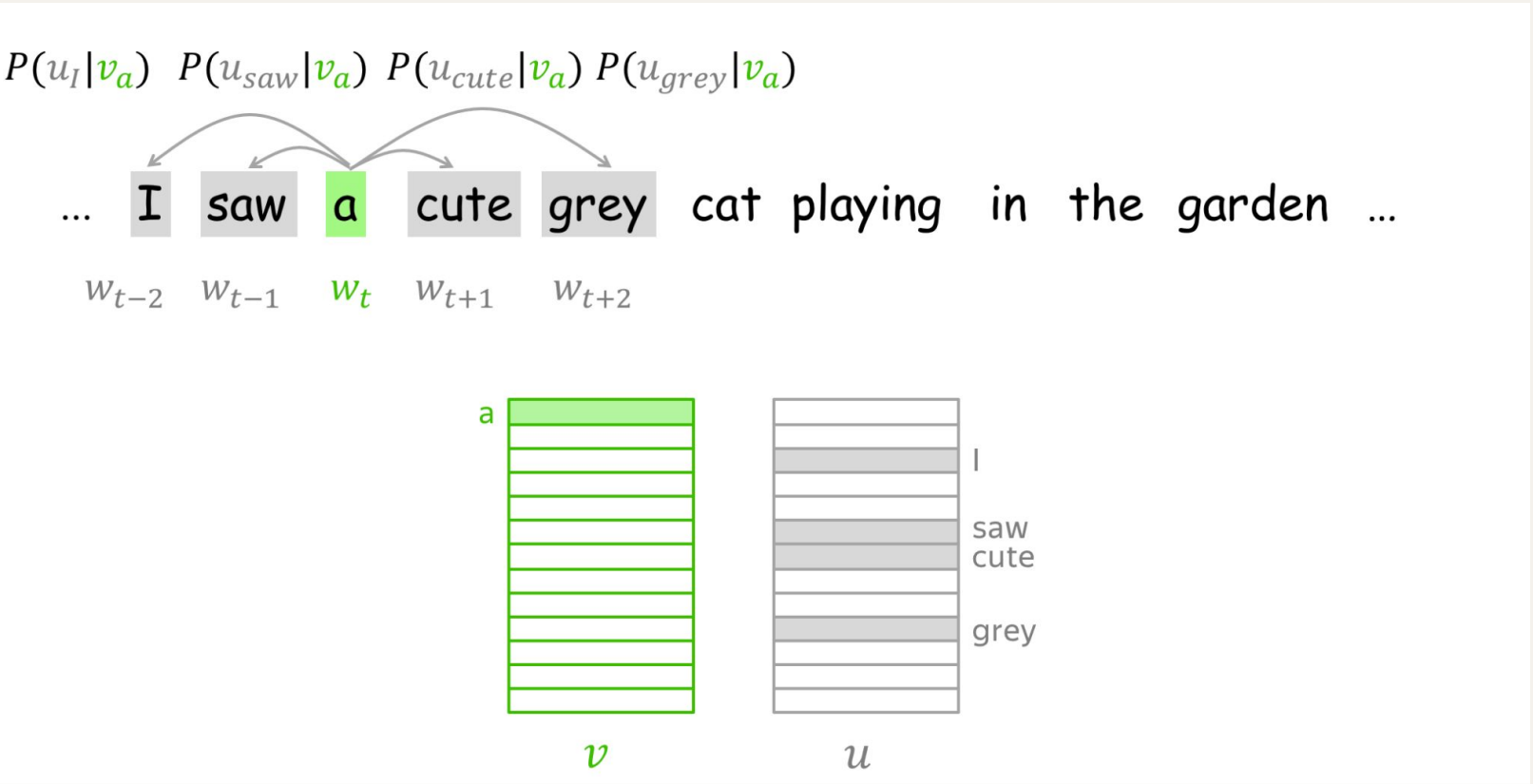
Dot product: measures similarity of o and c
Larger dot product = larger probability

Normalize over entire vocabulary to get probability distribution

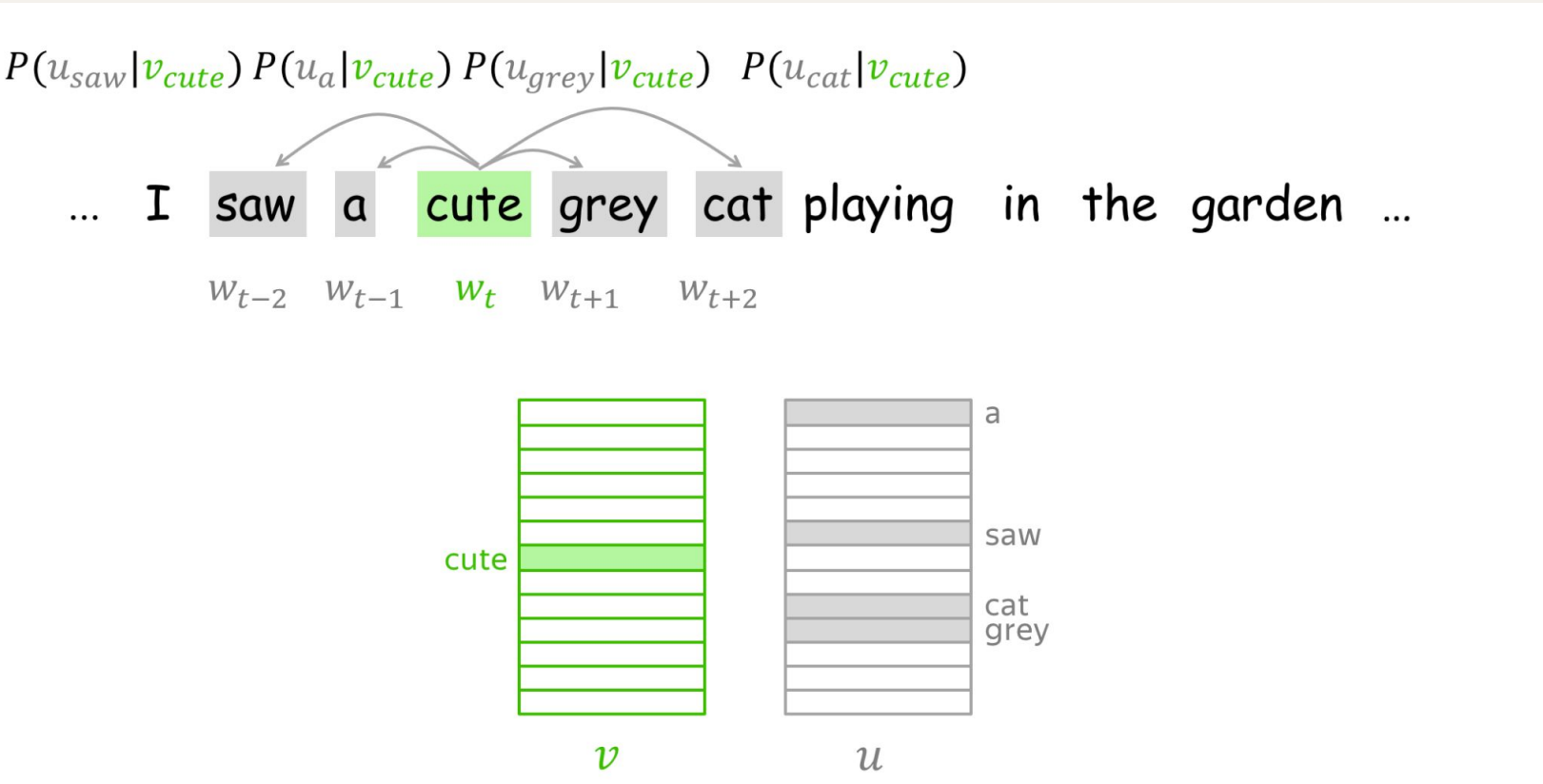
▶

Note: this is the softmax function! (click for the details)

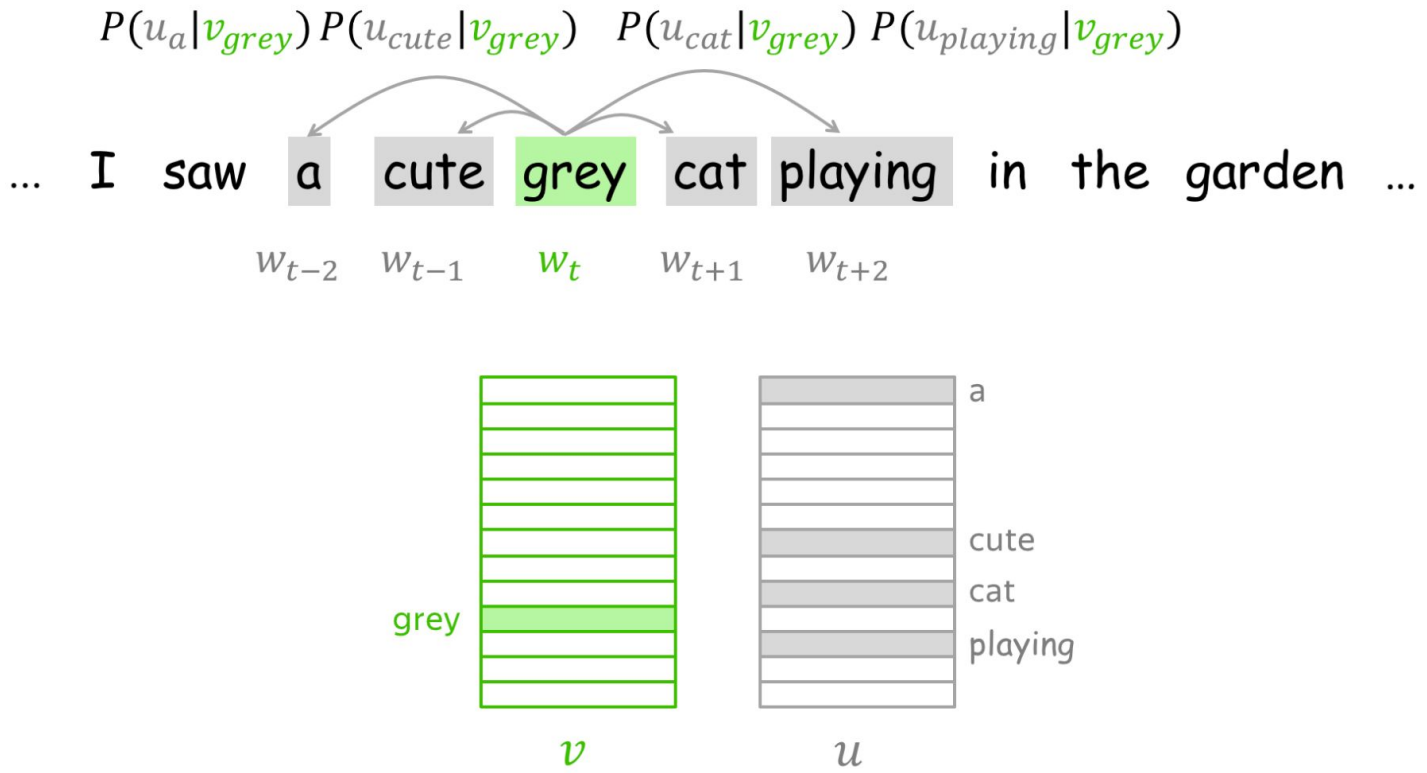
Word2Vec: A Prediction Based Method



Word2Vec: A Prediction Based Method

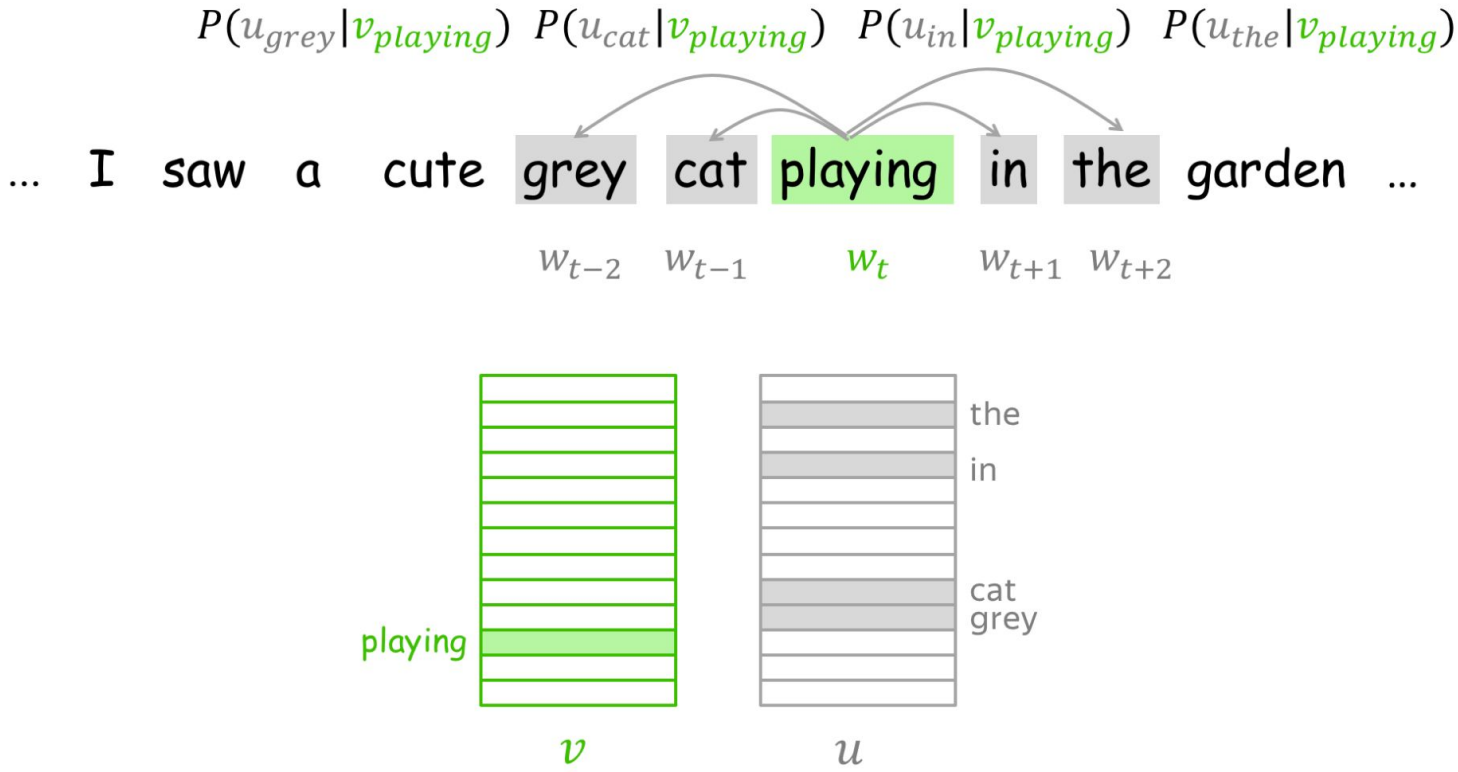


Word2Vec: A Prediction Based Method

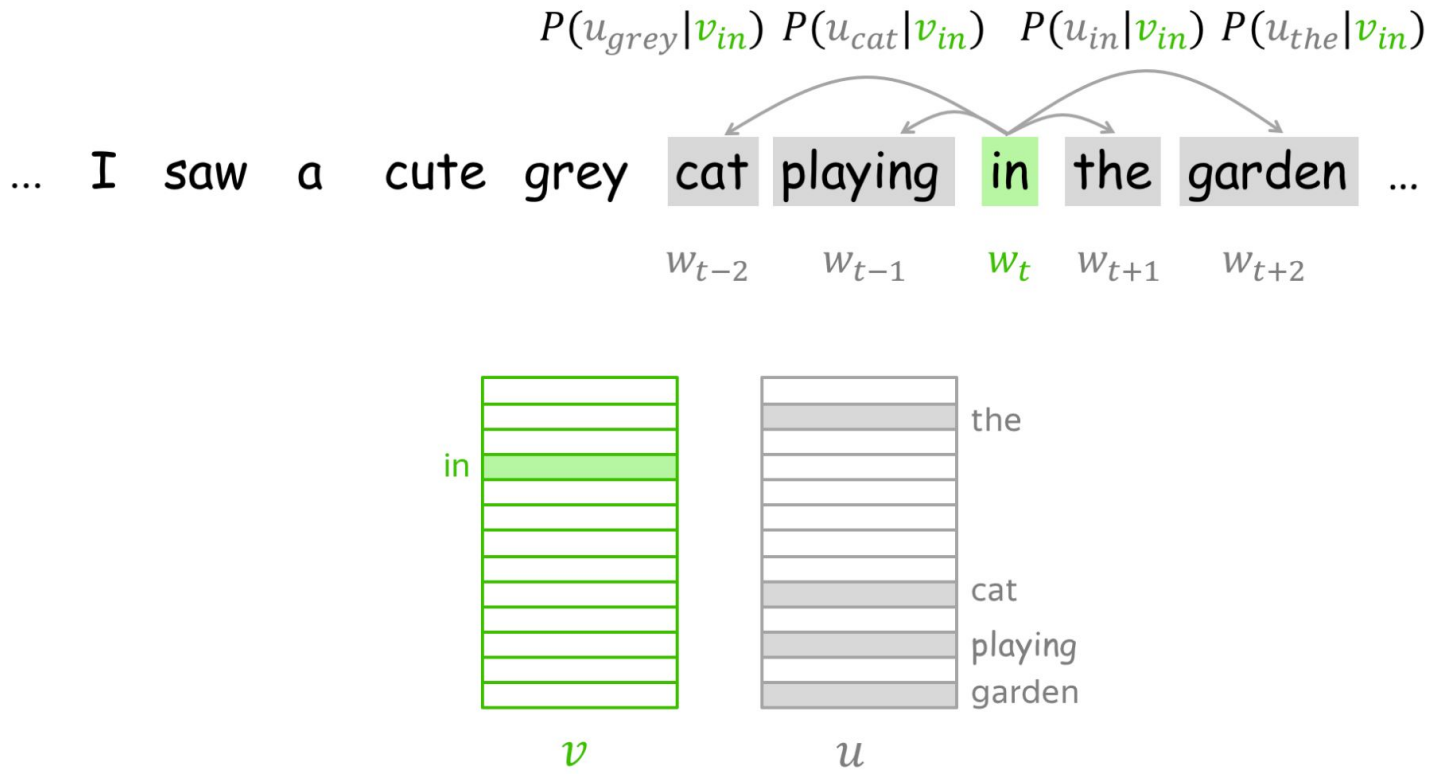


Word2Vec: A Prediction Based Method

Word2Vec: A Prediction Based Method



Word2Vec: A Prediction Based Method



How to train: by Gradient Descent, One Word at a Time

Let us recall that our parameters θ are vectors v_w and u_w for all words in the vocabulary. These vectors are learned by optimizing the training objective via gradient descent (with some learning rate α):

$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta).$$

One word at a time

We make these updates one at a time: each update is for a single pair of a center word and one of its context words. Look again at the loss function:

$$\text{Loss} = J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\le j\le m, j\neq 0}\log P(w_{t+j}|w_t,\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\le j\le m, j\neq 0} J_{t,j}(\theta).$$

For the center word w_t, the loss contains a distinct term $J_{t,j}(\theta) = -\log P(w_{t+j}|w_t,\theta)$ for each of its context words w_{t+j}. Let us look in more detail at just this one term and try to understand how to make an update for this step. For example, let's imagine we have a sentence

How to train: by Gradient Descent, One Word at a Time

... I saw a cute grey cat playing in the garden ...

with the central word cat, and four context words. Since we are going to look at just one step, we will pick only one of the context words; for example, let's take cute. Then the loss term for the central word cat and the context word cute is:

$$J_{t,j}(\theta) = -\log P(cute|cat) = -\log \frac{\exp u_{cute}^T v_{cat}}{\sum_{w \in Voc} \exp u_w^T v_{cat}} = -u_{cute}^T v_{cat} + \log \sum_{w \in Voc} \exp u_w^T v_{cat}.$$

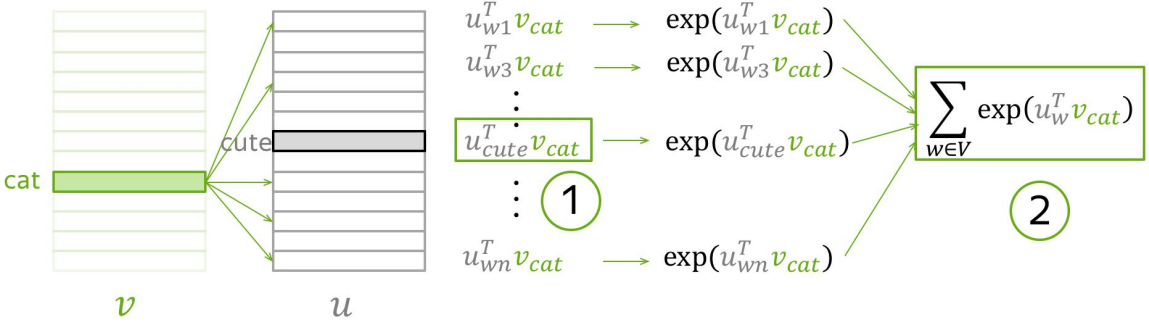
Note which parameters are present at this step:

- from vectors for central words, only v_{cat};
- from vectors for context words, all u_w (for all words in the vocabulary).

Only these parameters will be updated at the current step.

How to train: by Gradient Descent, One Word at a Time

1. Take dot product of v_{cat} with **all** u 2. exp 3. sum all



4. get loss (for this one step)

$$J_{t,j}(\theta) = \underbrace{-u_{cute}^T v_{cat}}_{1} + \underbrace{\log \sum_{w \in V} \exp(u_w^T v_{cat})}_{2}$$

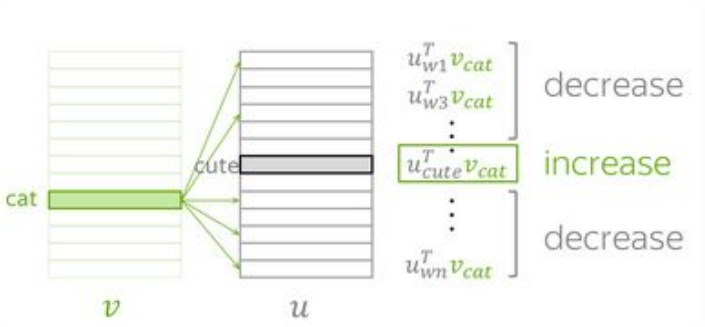
5. evaluate the gradient, make an update

$$v_{cat} := v_{cat} - \alpha \frac{\partial J_{t,j}(\theta)}{\partial v_{cat}}$$

$$u_w := u_w - \alpha \frac{\partial J_{t,j}(\theta)}{\partial u_w} \quad \forall w \in V$$

How to train: by Gradient Descent, One Word at a Time

By making an update to minimize $J_{t,j}(\theta)$, we force the parameters to increase similarity (dot product) of v_{cat} and u_{cute} and, at the same time, to decrease similarity between v_{cat} and u_w for all other words w in the vocabulary.

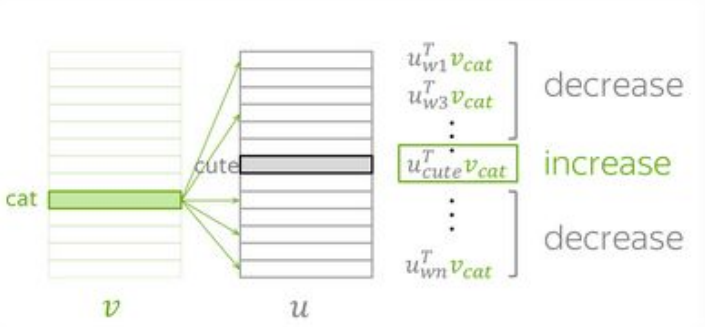


This may sound a bit strange: why do we want to decrease similarity between v_{cat} and all other words, if some of them are also valid context words (e.g., grey, playing, in on our example sentence)?

But do not worry: since we make updates for each context word (and for all central words in your text), on average over all updates our vectors will learn the distribution of the possible contexts.

How to train: by Gradient Descent, One Word at a Time

By making an update to minimize $J_{t,j}(\theta)$, we force the parameters to increase similarity (dot product) of v_{cat} and u_{cute} and, at the same time, to decrease similarity between v_{cat} and u_w for all other words w in the vocabulary.



This may sound a bit strange: why do we want to decrease similarity between v_{cat} and all other words, if some of them are also valid context words (e.g., grey, playing, in on our example sentence)?

But do not worry: since we make updates for each context word (and for all central words in your text), on average over all updates our vectors will learn the distribution of the possible contexts.

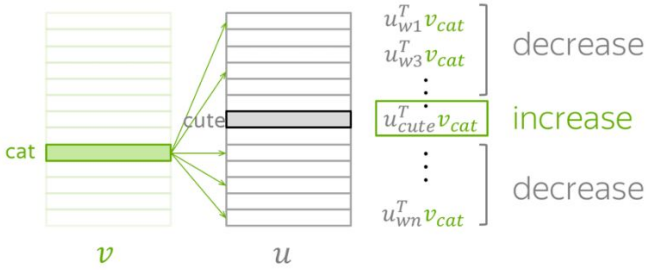
This method is extremely inefficient

Why?

Faster Training: Negative Sampling

Dot product of v_{cat}:

- with u_{cute} - increase,
- with **all other** u - decrease



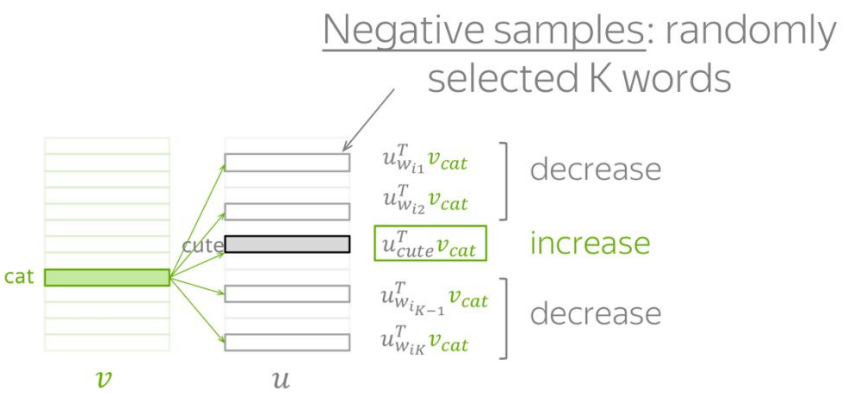
Parameters to be updated:

- v_{cat}
- u_w for all w in the vocabulary

|V| + 1 vectors

Dot product of v_{cat}:

- with u_{cute} - increase,
- with **a subset of other** u - decrease



Parameters to be updated:

- v_{cat}
- u_{cute} and u_w for w in K negative examples

K + 2 vectors

Faster Training: Negative Sampling

As before, we are increasing similarity between v_{cat} and u_{cute}. What is different, is that now we decrease similarity between v_{cat} and context vectors not for all words, but only with a subset of K "negative" examples.

Since we have a large corpus, on average over all updates we will update each vector sufficient number of times, and the vectors will still be able to learn the relationships between words quite well.

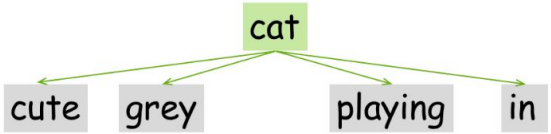
Formally, the new loss function for this step is:

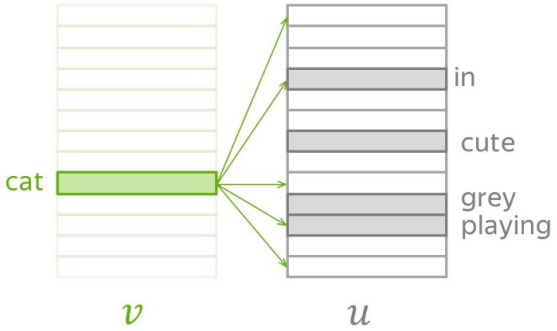
$$J_{t,j}(\theta) = -\log \sigma(u_{cute}^T v_{cat}) - \sum_{w \in \{w_{i_1}, \ldots, w_{i_K}\}} \log \sigma(-u_w^T v_{cat}),$$

where $w_{i_1}, \ldots, w_{i_K}$ are the K negative examples chosen at this step and $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function.

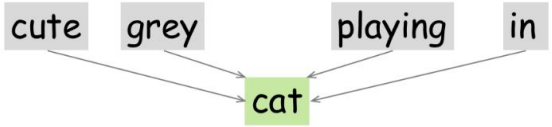
Word2Vec variants: Skip-Gram and CBOW

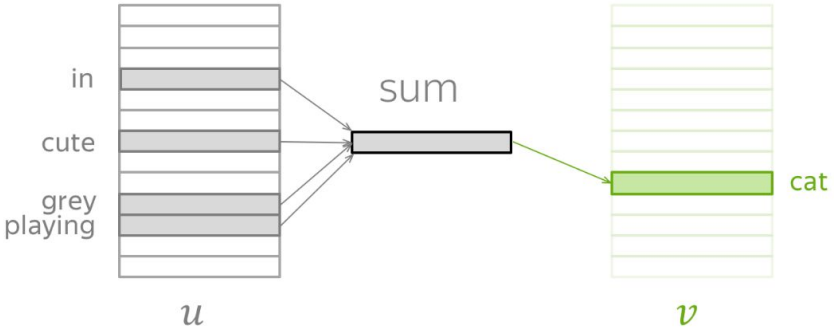
… I saw a cute grey cat playing in the garden …





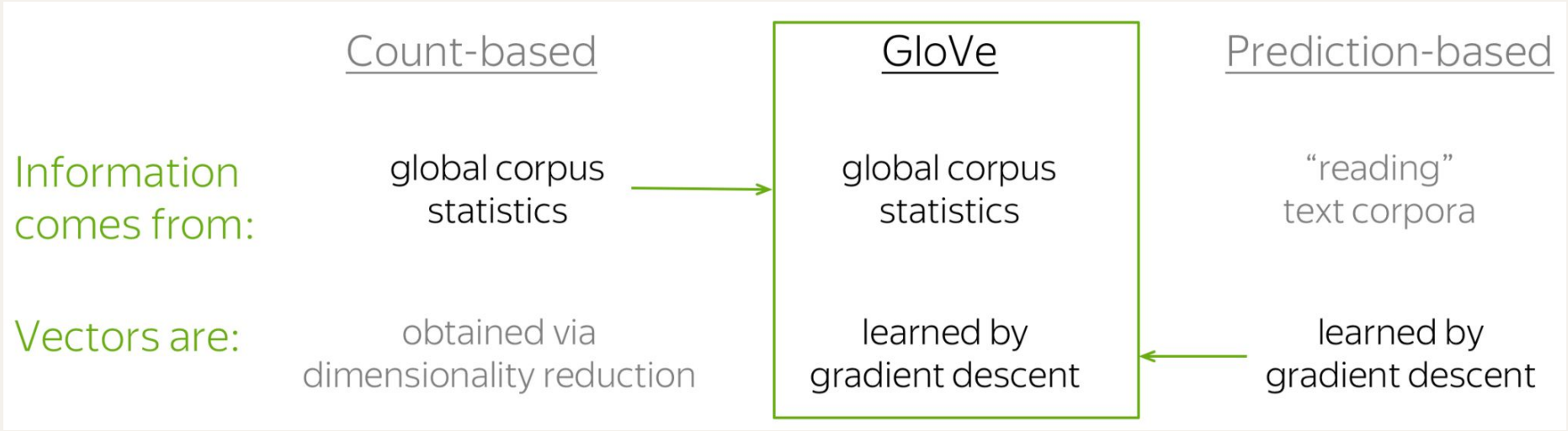
Skip-Gram: from central predict context (one at a time)





CBOW: from sum of context predict central

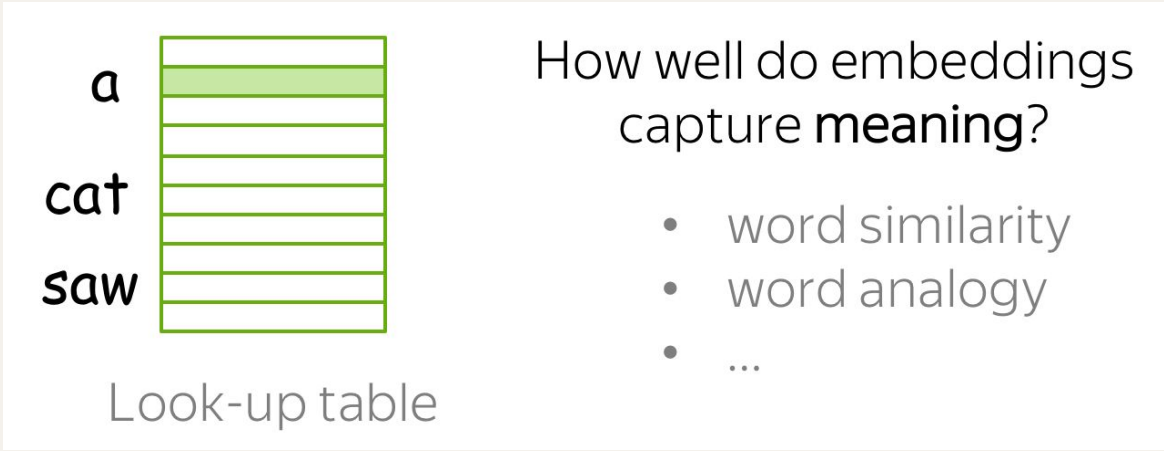
GloVe: Global Vectors for Word Representation



Evaluation of Word Embeddings

<u>Intrinsic Evaluation</u>: Based on Internal Properties

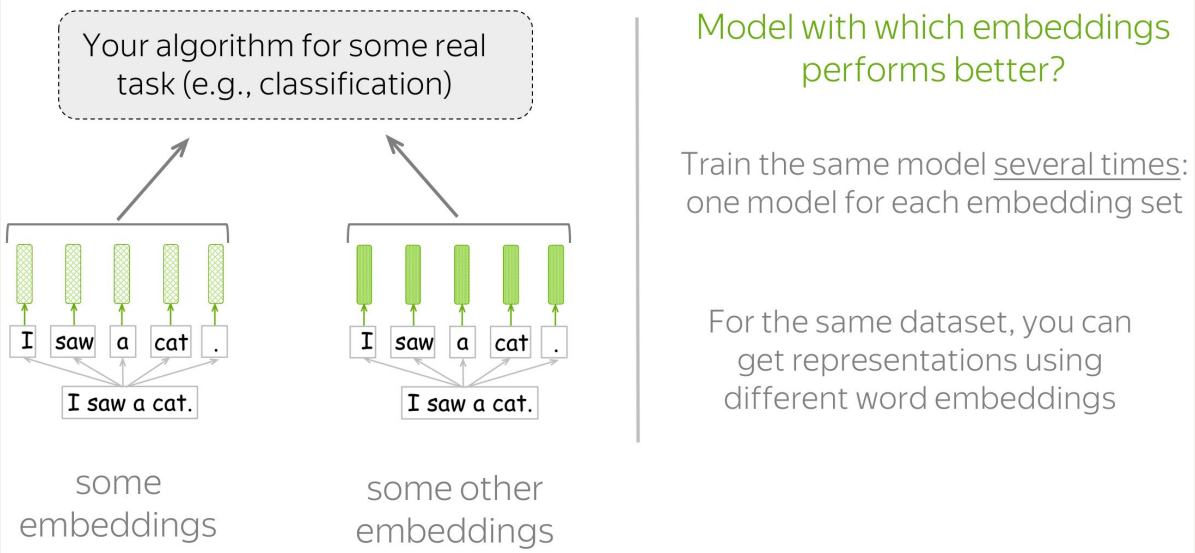
This type of evaluation looks at the internal properties of embeddings, i.e. how well they capture meaning.



Evaluation of Word Embeddings

Extrinsic Evaluation: On a Real Task

This type of evaluation tells which embeddings are better for the task you really care about (e.g., text classification, coreference resolution, etc.).



Evaluation of Word Embeddings

Intrinsic:

- usually fast
- does not tell what is better in practice

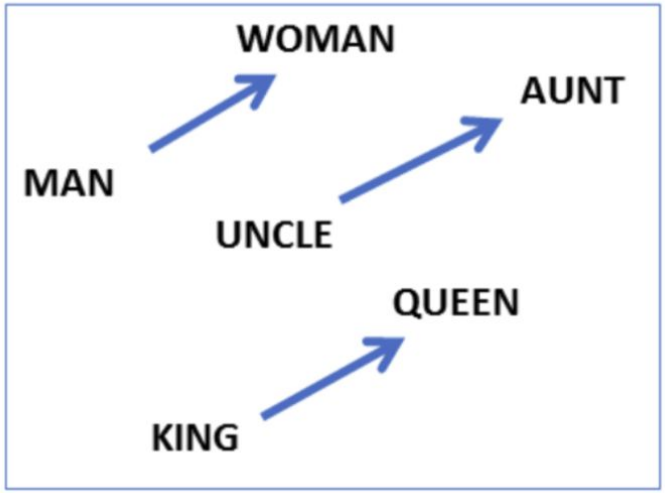
Extrinsic:

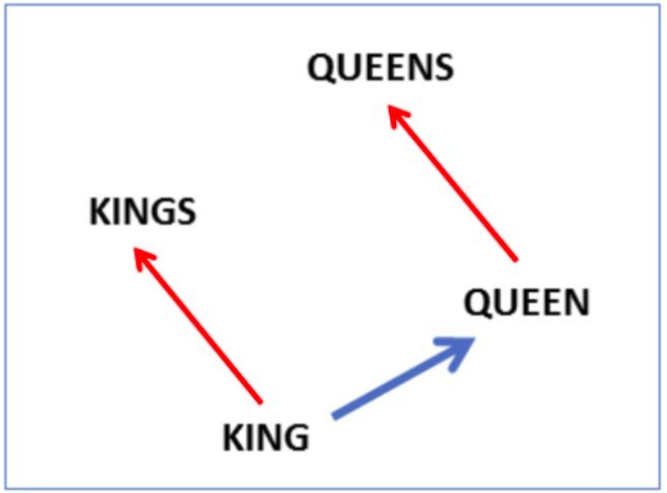
- tells directly what is better in practice
- training several real-task models is expensive

Take a Walk Through Space... Semantic Space!

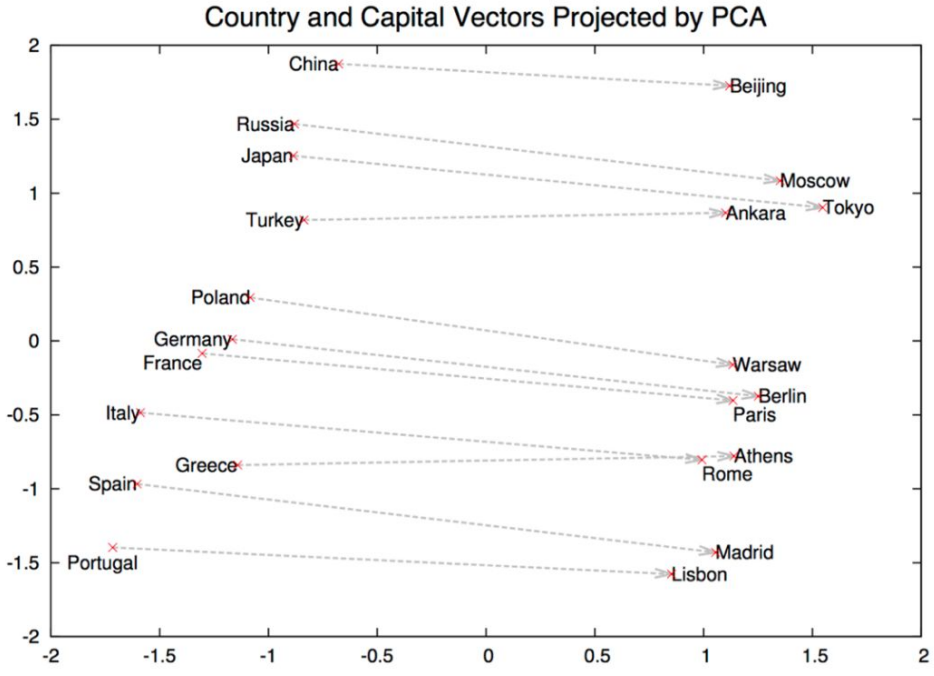
semantic: $v(\text{king}) - v(\text{man}) + v(\text{woman}) \approx v(\text{queen})$

syntactic: $v(\text{kings}) - v(\text{king}) + v(\text{queen}) \approx v(\text{queens})$

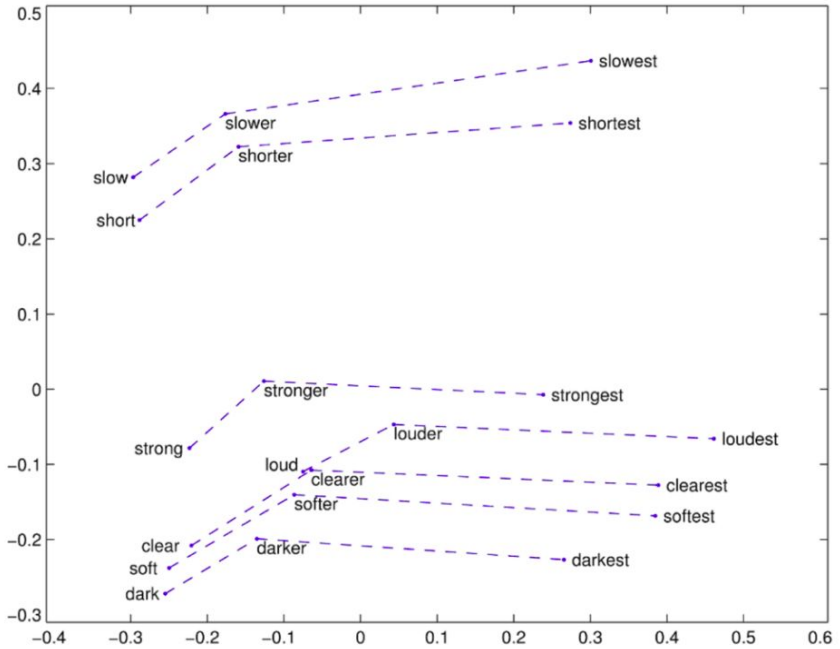




Take a Walk Through Space... Semantic Space!



Word2Vec



GloVe

Take a Walk Through Space... Semantic Space!

https://projector.tensorflow.org/

Analysis and Interpretability

Word Similarity Benchmarks

Correlation
between Human
and Model Score

word pair		score
vulgarism	profanity	9.62
subdividing	separate	8.67
friendships	brotherhood	7.5
exceedance	probability	5.0
assigned	allow	3.5
marginalize	interact	2.5
misleading	beat	1.25
radiators	beginning	0

Analysis and Interpretability

Word Analogy Benchmarks

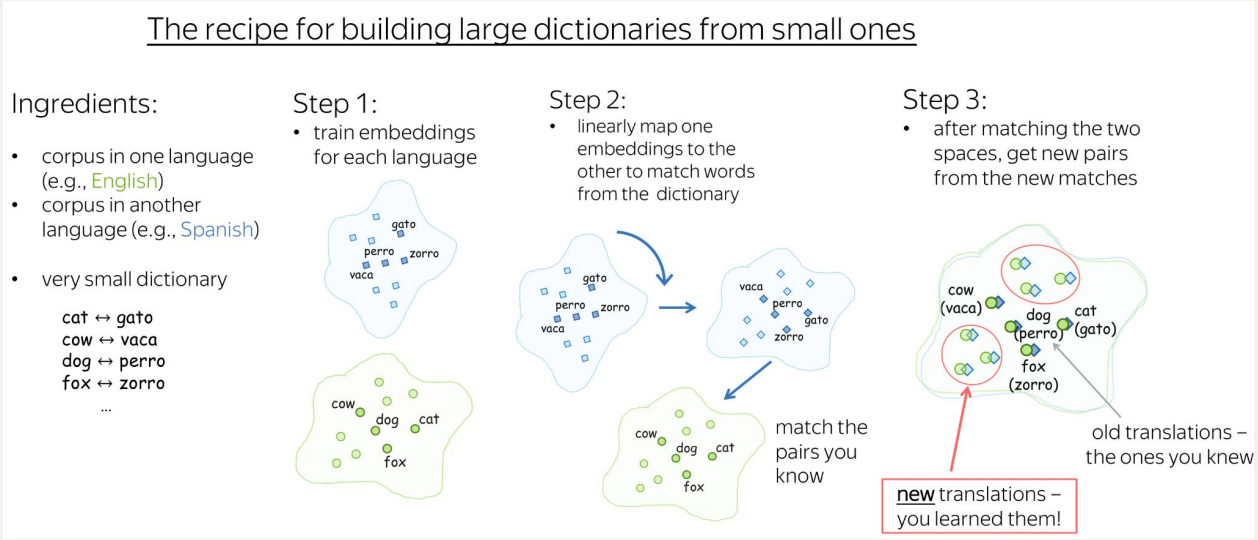
Analogy: **a** is to **a*** as **b** is to ___

Task: $v(a^*) - v(a) + v(b) \approx ?$ ⟶
- find the closest vector
- check if it corresponds to the correct word

relation	word pair 1		word pair 2	
man-woman	brother	sister	grandson	granddaughter
currency	Angola	kwanza	Iran	rial
opposite	possibly	impossibly	ethical	unethical
past tense	walking	walked	swimming	swam
superlative	easy	easiest	lucky	luckiest

Similarities across Languages

Turns out, relationships between semantic spaces are also (somewhat) linear: you can linearly map one semantic space to another so that corresponding words in the two languages match in the new, joint semantic space.



Similarities across Languages

[1710.04087] Word Translation Without Parallel Data

Recommended Resources

Recommended Reading

- Jurafsky - Chapter 2
- Lena Voita - Word Embeddings
- Jurafsky - Lecture 3 - Vector Semantics
- Previous post Understanding Pointwise Mutual Information in Statistics