word2vec

intuition, math, and model

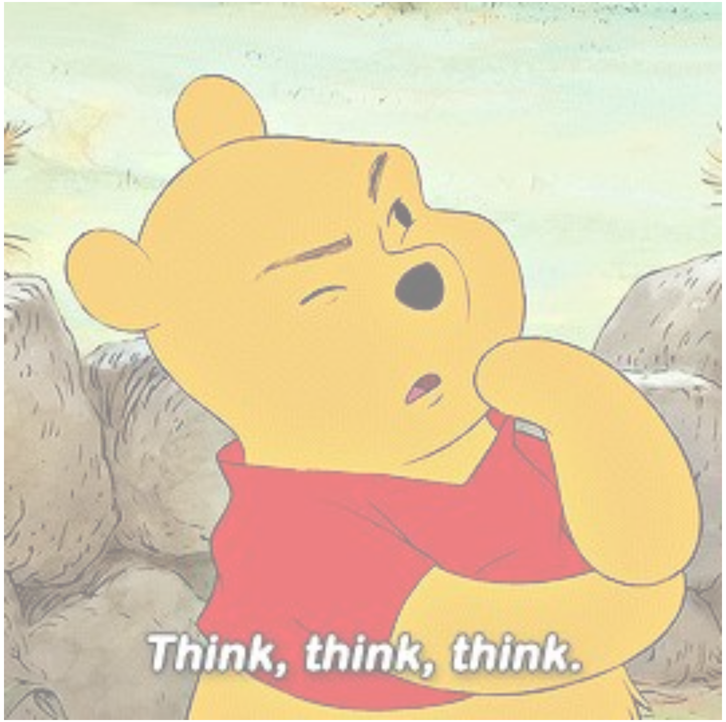
How to learn a word's meaning

Remember our example?

What does the word **tezgüino** mean?

Examples how it's used in a sentences:

1. A bottle of **tezgüino** is on the table.
2. Everyone likes **tezgüino**.
3. **Tezgüino** makes you drunk.
4. We make **tezgüino** out of corn.



How to learn a word's meaning

Remember our example?

What does the word **tezgüino** mean?

Examples how it's used in a sentences:

1. A bottle of **tezgüino** is on the table.
2. Everyone likes **tezgüino**.
3. **Tezgüino** makes you drunk.
4. We make **tezgüino** out of corn.

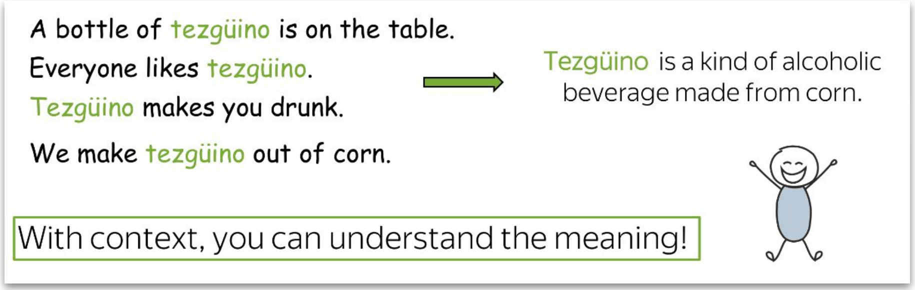
Implication

- Words with **similar meaning** appear in **similar context** (word windows or sentences)
- To **capture a word's meaning** with numbers, its *numeric representation* should summarize in which word contexts it occurs

Word embedding methods

Intuition

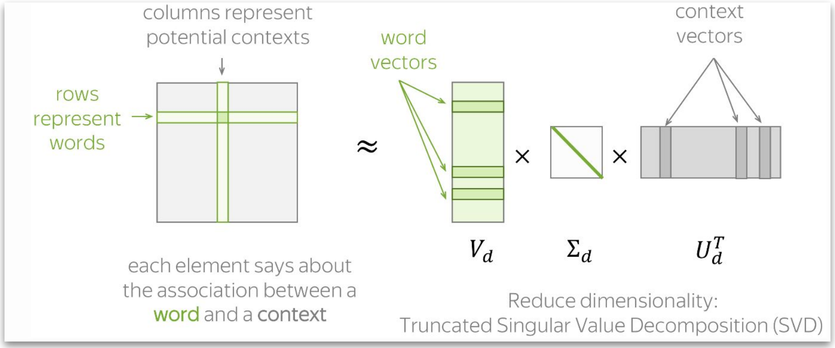
- co-occurrence patterns and/or word context information summarizes a word's meaning and functions
- embedding methods condense these distributional patterns into low-dimensional vectors



Word embedding methods

Count-based methods

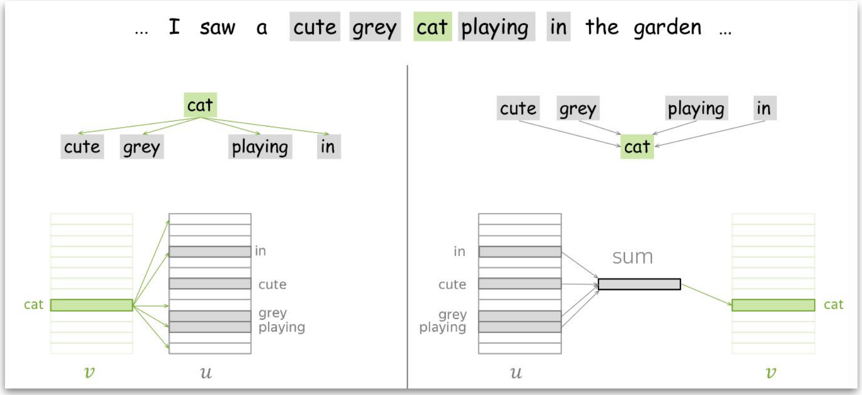
learn word vectors by reducing the dimensionality of word context representations



our focus today: **word2vec**

Prediction-based methods

learn good word vectors by predicting their context (or *vice versa*)



Intuition

learning word embeddings
by predicting words’ context

Focus words and context words

Implication of *fill-the-blank* example

> *To* ***capture a word's meaning*** *with numbers, its* numeric representation *should summarize in which word contexts it occurs*

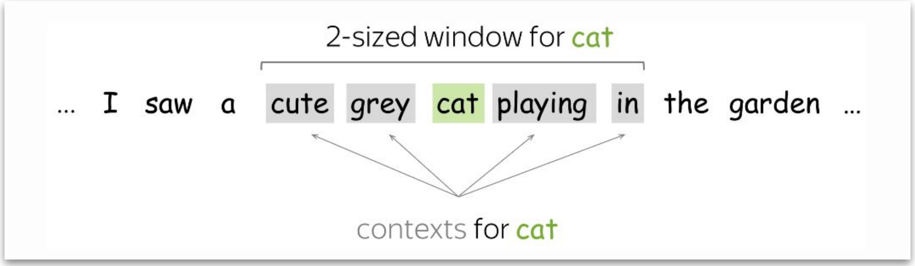
Focus words and context words

Implication of *fill-the-blank* example

> *To **capture a word's meaning** with numbers, its* numeric representation *should summarize in which word contexts it occurs*

Approach

Use a word's embedding to predict which words occur in its context
⇒ *incentive* to learn about its usage



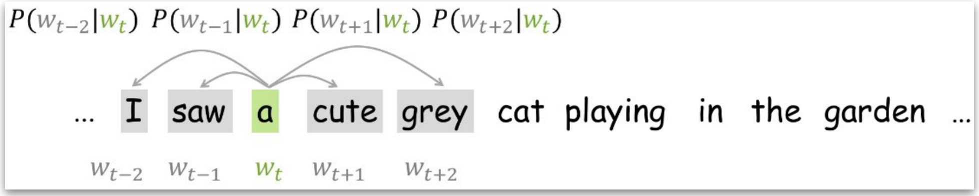
Notation

- **focus word**: word whose meaning we want to capture
- **context word**: word occurring in a window around the focus word's

The skip-gram algorithm

Use a word's embedding to predict which words occur in its context

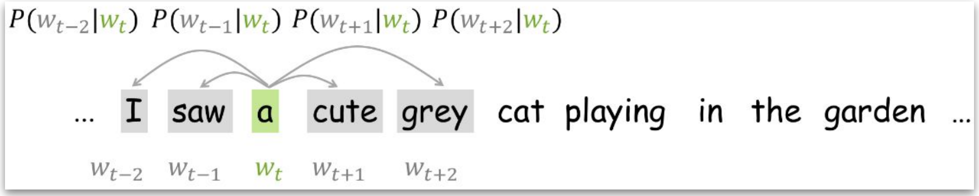
- w_i is the word *i*'s embedding
- ***t*** indicates a word's position relative to the focus word
- *P*(*a* | *b*) is the ***conditional probability*** that ***a*** occurs given ***b***

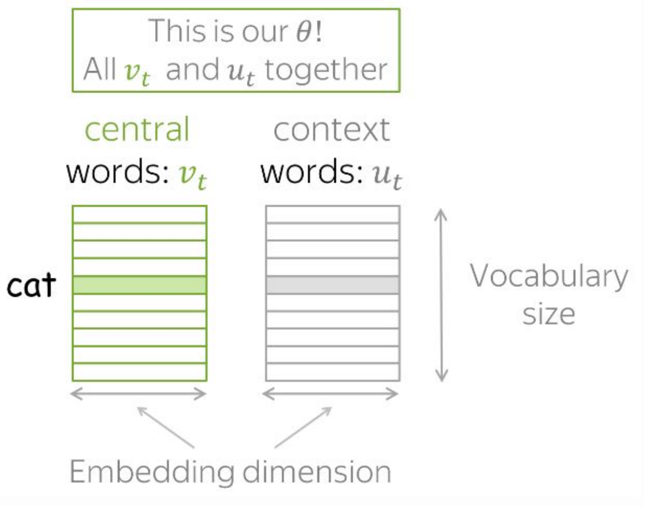


The skip-gram algorithm

Use a word's embedding to predict which words occur in its context

- w_i is the word *i*'s embedding
- ***t*** indicates a word's position relative to the focus word
- *P*(*a* | *b*) is the ***conditional probability*** that ***a*** occurs given ***b***
- using different vectors depending on whether the word is a focus or context word

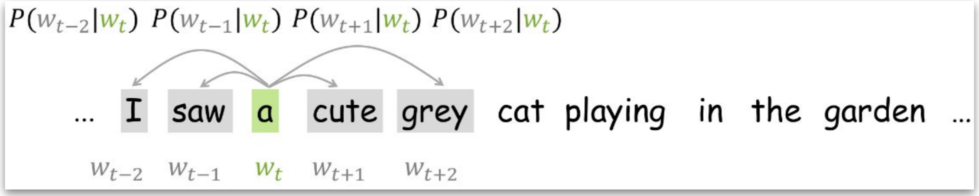




The skip-gram algorithm

Use a word's embedding to predict which words occur in its context

- w_i is the word *i*'s embedding
- ***t*** indicates a word's position relative to the focus word
- *P*(*a* | *b*) is the ***conditional probability*** that ***a*** occurs given ***b***
- using different vectors depending on whether the word is a focus or context word



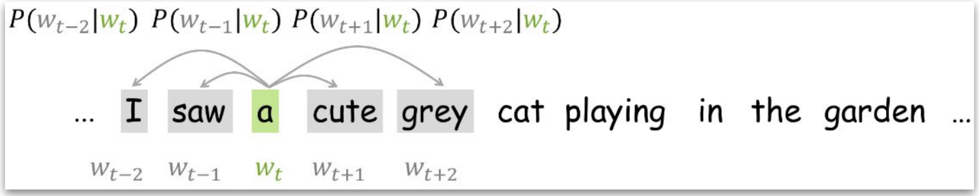
Overall, we want to ***maximize***

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

The skip-gram algorithm

Use a word's embedding to predict which words occur in its context

- w_i is the word *i*'s embedding
- t indicates a word's position relative to the focus word
- $P(a \mid b)$ is the ***conditional probability*** that *a* occurs given *b*



Overall, we want to ***maximize***

focus word

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

positions in the text corpus

relative position to focus word

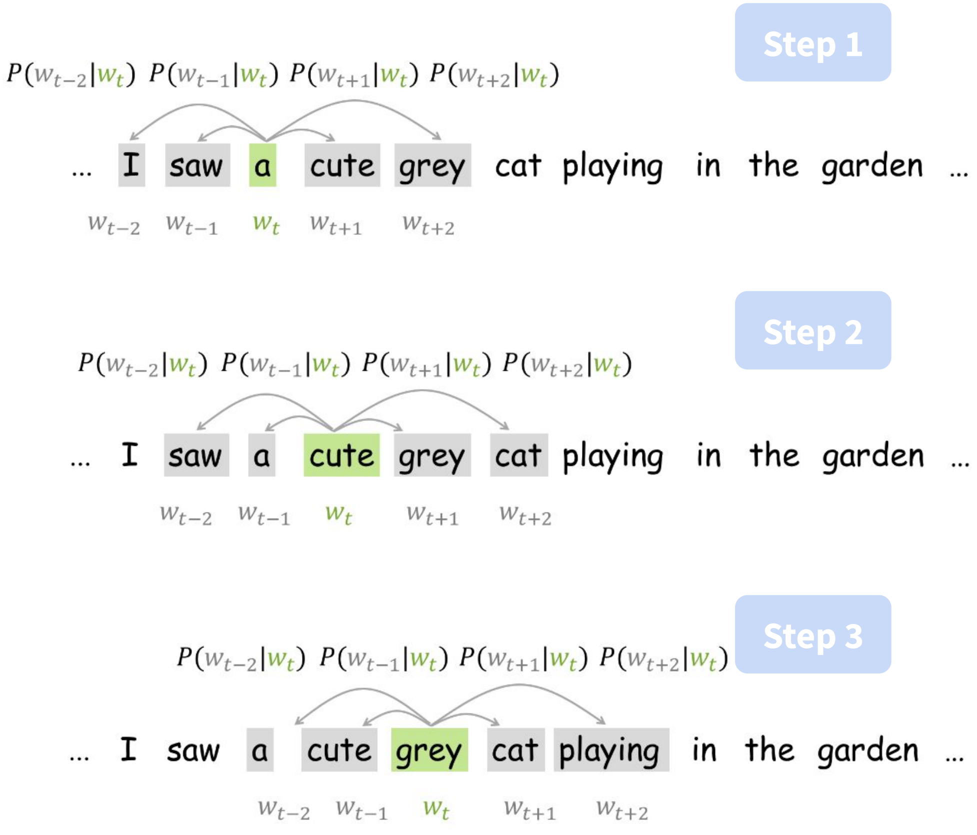
the "embeddings" we optimize

One step at a time

The skip-gram algorithm

Use a word's embedding to predict which words occur in its context

- slide over all *T* locations in a corpus
- predict words in focus word's "neighborhood" of ±*m* words
  (*m* is called window size)



The skip-gram algorithm

Use a word's embedding to predict which words occur in its context

- w_i is the word *i*'s embedding
- ***t*** indicates a word's position relative to the focus word
- $P(a \mid b)$ is the ***conditional probability*** that ***a*** occurs given ***b***
- using different vectors depending on whether the word is a focus or context word

Math

we measure $P(\mathbf{c} \mid \mathbf{v})$, the probability of observing context words *c* given focus word *v*, using the similarity of word ***c*** and *v*'s embeddings:

$$P(c \mid v) \propto \mathrm{sim}(w_c, w_v)$$

Intuition *c* and *v*'s embeddings need to be similar to predict a high $P(c \mid v)$

The skip-gram algorithm

Problem

to get a probability estimate for all words that might occur in the focus words' context, we need to comput **sim**(w_c, w_v) for the entire vocabulary

Math

we measure $P(\mathbf{c} \mid \mathbf{v})$, the probability of observing context words c given focus word v, using the similarity of word c and v's embeddings:

$$P(c \mid v) \propto \mathbf{sim}(w_c, w_v)$$

The skip-gram algorithm

Problem

to get a probability estimate for all words that might occur in the focus words' context, we need to comput **sim**(w_c, w_v) for the entire vocabulary

… at every position …

⇒ computationally too demanding

Math

we measure $P(\mathbf{c} \mid \mathbf{v})$, the probability of observing context words c given focus word v, using the similarity of word c and v's embeddings:

$$P(c \mid v) \propto \mathbf{sim}(w_c, w_v)$$

The skip-gram algorithm

focus	***context***	***label***
cute	saw	“hit”
cute	a	“hit”
cute	grey	“hit”
cute	cat	“hit”

Solution ⇒ “negative sampling”

1. take actual target words (label = “hit”)

The skip-gram algorithm

focus	*context*	*label*
cute	saw	"hit"
cute	a	"hit"
cute	grey	"hit"
cute	cat	"hit"
cute	do	"miss"
cute	melon	"miss"
cute	tezgüino	"miss"
…	…	…

Solution ⇒ "negative sampling"

1. take actual target words (label = "hit")
2. take a random sample of words from the vocabulary (label = "miss")

The skip-gram algorithm

focus	*context*	*label*	*sim*
cute	saw	“hit”	0.23
cute	a	“hit”	0.41
cute	grey	“hit”	0.33
cute	cat	“hit”	0.68
cute	do	“miss”	0.10
cute	melon	“miss”	-0.12
cute	tezgüino	“miss”	-0.43
…	…	…	

Solution ⇒ “negative sampling”

1. take actual target words (label = “hit”)
2. take a random sample of words from the vocabulary (label = “miss”)
3. compute **sim**(w_c, w_v) for “hits” and “misses” with focus word

The skip-gram algorithm

focus	***context***	***label***	***sim***
cute	saw	“hit”	0.23
cute	a	“hit”	0.41
cute	grey	“hit”	0.33
cute	cat	“hit”	0.68
cute	do	“miss”	0.10
cute	melon	“miss”	-0.12
cute	tezgüino	“miss”	-0.43
…	…	…	

Solution ⇒ “negative sampling”

1. take actual target words (label = “hit”)
2. take a random sample of words from the vocabulary (label = “miss”)
3. compute **sim**(w_c, w_v) for “hits” and “misses” with focus word
4. **classify** if context word is “hit” or “miss”

The skip-gram algorithm

focus	***context***	***label***	***sim***
cute	saw	“hit”	☝️ 0.23
cute	a	“hit”	☝️ 0.41
cute	grey	“hit”	☝️ 0.33
cute	cat	“hit”	☝️ 0.68
cute	do	“miss”	👇 0.10
cute	melon	“miss”	👇 -0.12
cute	tezgüino	“miss”	👇 -0.43
…	…	…	

Solution ⇒ “negative sampling”

1. take actual target words (“hits”)
2. take a random sample of words from the vocabulary (“misses”)
3. compute **sim**(w_c, w_v) for “hits” and “misses” with focus word
4. classify if context word is “hit” or “miss”
5. **update** model parameters $\boldsymbol{\theta}$ (our word embeddings) to
    - increase probability of “hits” ☝️
    - decrease probability of “misses” 👇

   using “**gradient descent**”

The skip-gram algorithm

focus	*context*	*label*	*sim*
cute	saw	“hit”	0.23
cute	a	“hit”	0.41
cute	grey	“hit”	0.33
cute	cat	“hit”	0.68
cute	do	“miss”	0.10
cute	a	“miss”	0.41
cute	tezgüino	“miss”	-0.43
…	…	…	

Solution ⇒ “negative sampling”

1. take actual target words (“hits”)
2. take a random sample of words from the vocabulary (“misses”)
   tiny chance that this random sampling of negative words causes “**label noise**”
   ⇒ but inconsequential overall ✔

That’s it

If you want to learn more about word2vec and other algorithms (from light to dense)

- Lena Voita’s NLP online short-course
- Stanford CS224N (2021) lecture on word2vec
- Jurafsky & Martin, Chapter 6

- motivation: simple idea, clever algorithm, illustrates lots of key ideas in deep learning-based NLP
- building blocks
    - target/focal/focus words and context/neighboring words
    - self-supervised learning
    - skipgram and CBOW
        - CBOW: predict target word from its context.
            - goal: max. probability of the target word given its context
            - implementation: sum/average context words' embeddings into 1d vector; use it to predict target word (as in nominal regression).
        - skip-gram: predict context words given target word.
            - goal: max. probability of context words given target word
            - take one context word at a time; an predict it given the target word's vector
    - Pr(y | **x**, *ß*)
    - predicting target/context word as classification task (with large label space)
        - costly
        - like a nominal/categorical regression, but need non-linearities for learning "good" word embeddings
    - softmax and probability distribution over the vocabulary
    -
    - negative sampling
    - *stochatstic gradient decent* and back propagation (show explainer video)

Social Science Applications

- for measurement purposes
-
- features in downstream tasks

Different ways of using word embeddings in CSS research

1. as primary quantity of interest
    a. to compute associations (Kozlowski, WEAT)
    b.
2. as a tool
    a. scale documents (e.g., Gennaro and Ash, 2022)
    b. to compare language use (Rodriguez, Spirling, and Stewart)