

NLP - Section 6

Word Vectors

Embeddings

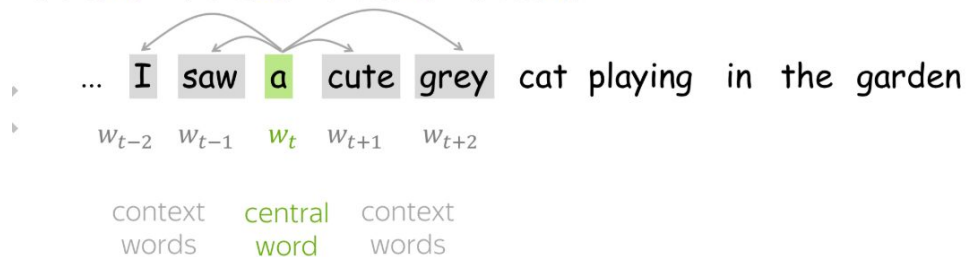
- An embedding is a numerical representation of a word.
- The representation of the word doesn't account for the context.
- Words that occur in the same context have close numerical representation
- Two types:
 - A. Word Embeddings: no context
 - B. Contextual Embeddings: with context



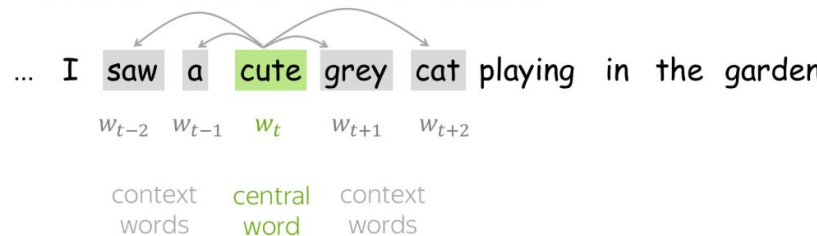
Word2Vec

- There are various techniques for word embeddings, one of which is Word2Vec
- Word2Vec is based on continuous skip-gram architecture where it predicts the context words given the current words.

$$P(w_{t-2}|w_t) \quad P(w_{t-1}|w_t) \quad P(w_{t+1}|w_t) \quad P(w_{t+2}|w_t)$$



$$P(w_{t-2}|w_t) \quad P(w_{t-1}|w_t) \quad P(w_{t+1}|w_t) \quad P(w_{t+2}|w_t)$$



Word2Vec Cont.

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

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

agrees with our
plan above



go over text



with a sliding
window

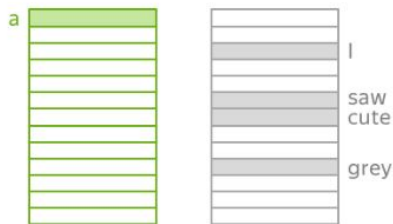
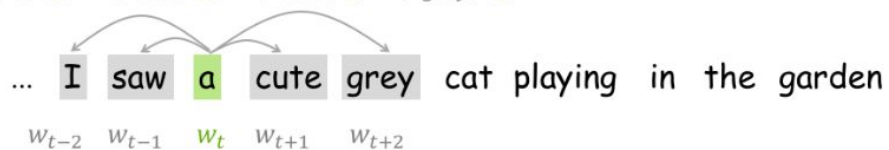


compute probability of the
context word given the central

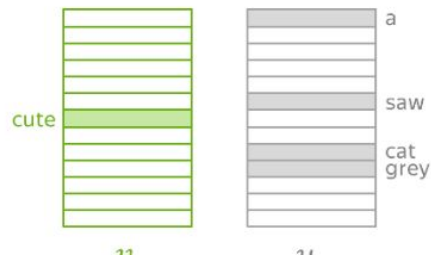
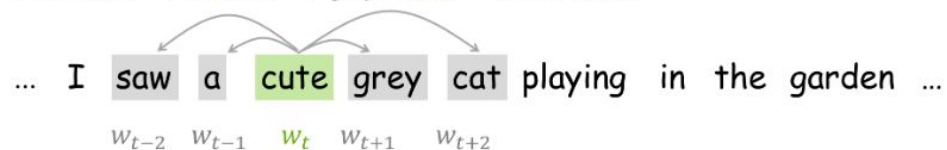


Skip Gram

$$P(u_I | v_a) \quad P(u_{\text{saw}} | v_a) \quad P(u_{\text{cute}} | v_a) \quad P(u_{\text{grey}} | v_a)$$



$$P(u_{\text{saw}} | v_{\text{cute}}) \quad P(u_a | v_{\text{cute}}) \quad P(u_{\text{grey}} | v_{\text{cute}}) \quad P(u_{\text{cat}} | v_{\text{cute}})$$



Word2Vec Calculations

how to calculate this

The diagram illustrates the calculation of the Word2Vec loss function. The equation is:
$$\text{Loss} = J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-m \leq j \leq m, \\ j \neq 0}} \log P(w_{t+j} | w_t, \theta)$$
 Annotations include: 'agrees with our plan above' pointing to the loss function; 'go over text' pointing to the summation over t ; 'with a sliding window' pointing to the summation over j ; and 'compute probability of the context word given the central' pointing to the probability term $\log P(w_{t+j} | w_t, \theta)$. A large grey arrow points from the text 'how to calculate this' to the boxed probability term.

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

- We use softmax function

The diagram shows the softmax function for word probability:
$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$
 Annotations include: 'Dot product: measures similarity of o and c . Larger dot product = larger probability' pointing to the numerator $\exp(u_o^T v_c)$; and 'Normalize over entire vocabulary to get probability distribution' pointing to the denominator $\sum_{w \in V} \exp(u_w^T v_c)$.

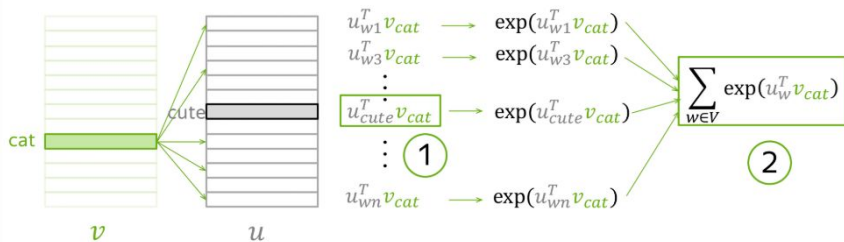
Dot product: measures similarity of o and c . Larger dot product = larger probability

Normalize over entire vocabulary to get probability distribution

Gradient Descent Training

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

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



4. get loss (for this one step)
5. evaluate the gradient, make an update

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

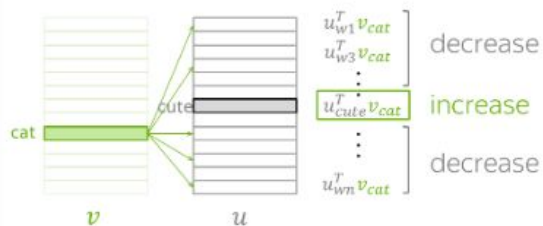
$$v_{\text{cat}} := v_{\text{cat}} - \alpha \frac{\partial J_{t,j}(\theta)}{\partial v_{\text{cat}}}$$

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

Negative Sampling

Dot product of v_{cat} :

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



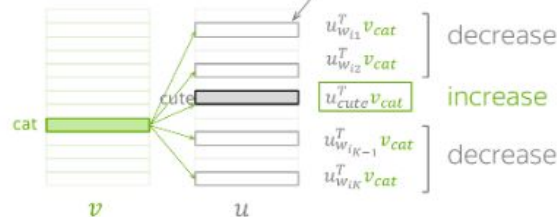
Parameters to be updated:

- 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

Negative samples: randomly selected K words



Parameters to be updated:

- v_{cat}
 - u_{cute} and u_w for w in K negative examples
- $K + 2$ vectors

Negative Sampling Cont.

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

Note that $\sigma(-x) = \frac{1}{1+e^x} = \frac{1 \cdot e^{-x}}{(1+e^x) \cdot e^{-x}} = \frac{e^{-x}}{1+e^{-x}} = 1 - \frac{1}{1+e^x} = 1 - \sigma(x)$. Then the loss can also be written as:

$$J_{t,j}(\theta) = -\log \sigma(u_{cute}^T v_{cat}) - \sum_{w \in \{w_{i_1}, \dots, w_{i_K}\}} \log(1 - \sigma(u_w^T v_{cat})).$$



Training

1. We initialize the matrices for the embeddings and the context. Both have the same (vocab_size, embedding_size)
2. We create a dataset with 0 and 1, 0 is not in context and 1 is in context of the center word
3. We then train the model using gradient descent to predict if the word is in context or not of the center, this spans the entire corpus.



Code

- Word2Vec Implementation from Scratch

