Natural Language Processing with Deep Learning

Lecture 6 — Text classification 3: Learning word embeddings

Prof. Dr. Ivan Habernal

November 17, 2023

PADERBORN UNIVERSITY

Natural Language Processing Group
Paderborn University
We focus on Trustworthy Human Language Technologies

www.trusthlt.org

Motivation

I give you a large corpus or plain text data

Can you build a model that will answer any of the ‘word analogy’ tasks?

- ‘Germany to Berlin is like France to ?’
- ‘Man to king is like woman to ?’

Recap: Neural LMs learn word embeddings

Recap: Neural LMs learn word embeddings

We need to talk about the dot product

Distributional hypothesis

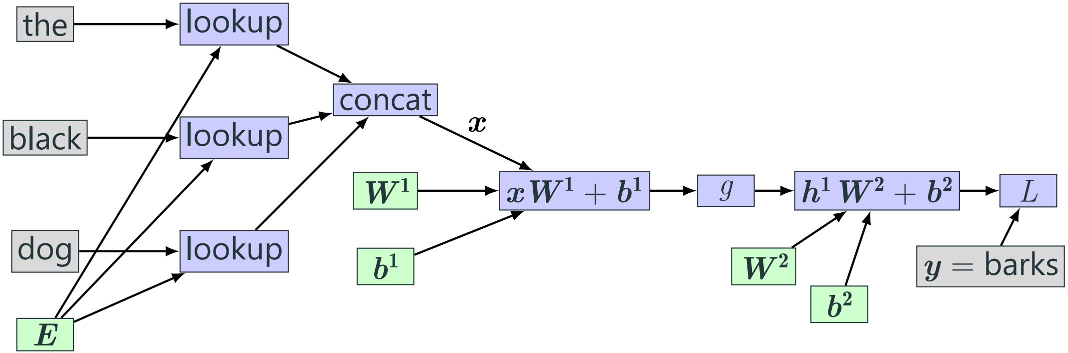
From neural LMs to training word embeddings

word2vec

FastText embeddings: Sub-word embeddings

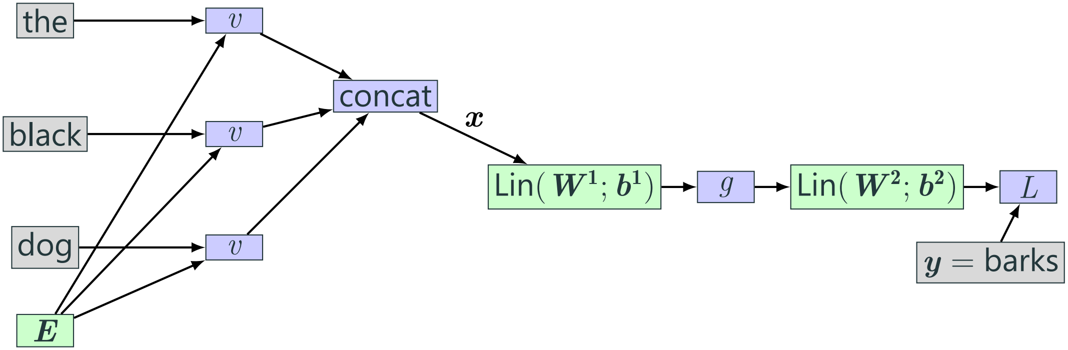
Advantages and limitations of words embeddings

Neural language model, context = 3 preceding words



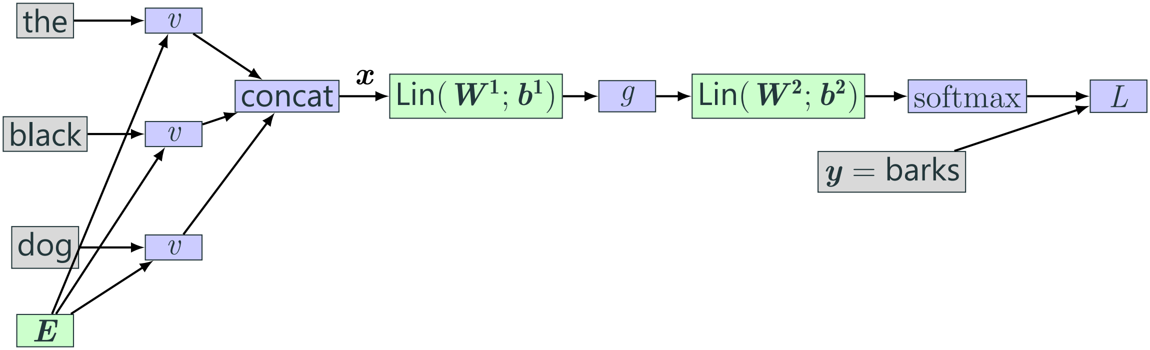
Recall: Green nodes = trainable parameters $\boldsymbol{\Theta}$, blue nodes = differentiable functions, gray nodes = constants

Simplify notation: Lookup v, Linear layers incl. parameters



Green nodes = trainable parameters or nodes with trainable parameters $\boldsymbol{\Theta}$, blue nodes = differentiable functions, gray nodes = constants

We forgot softmax for actually predicting distribution over V



the, black, dog, $\boldsymbol{y}$ = one-hot encoding, $\mathbb{R}^{|V|}$

Loss L and gold label $\boldsymbol{y}$ — only for training

We need to talk about the dot product

Recap: Neural LMs learn word embeddings

We need to talk about the dot product

Distributional hypothesis

From neural LMs to training word embeddings

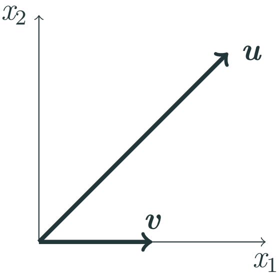
word2vec

FastText embeddings: Sub-word embeddings

Advantages and limitations of words embeddings

Geometry of dot product

For two n-dimensional vectors $\boldsymbol{u}$ and $\boldsymbol{v}$

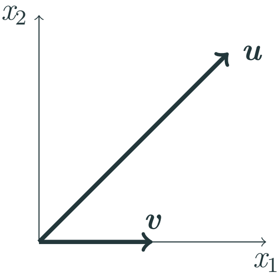


Geometry of dot product

For two n-dimensional vectors $\boldsymbol{u}$ and $\boldsymbol{v}$

Algebraic

$$\boldsymbol{u} \cdot \boldsymbol{v} = \sum_{i=1}^{n} \boldsymbol{u}_{[i]} \boldsymbol{v}_{[i]}$$



Geometry of dot product

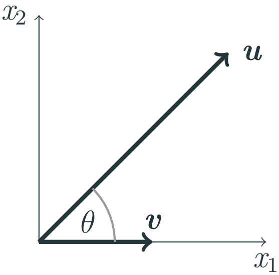
For two n-dimensional vectors $\boldsymbol{u}$ and $\boldsymbol{v}$

Algebraic

$$\boldsymbol{u} \cdot \boldsymbol{v} = \sum_{i=1}^{n} \boldsymbol{u}_{[i]} \boldsymbol{v}_{[i]}$$

Geometric

$$\boldsymbol{u} \cdot \boldsymbol{v} = \|\boldsymbol{u}\| \|\boldsymbol{v}\| \cos(\theta)$$



Geometry of dot product

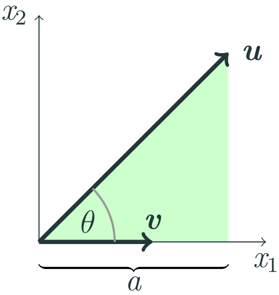
For two n-dimensional vectors $\boldsymbol{u}$ and $\boldsymbol{v}$

Algebraic

$$\boldsymbol{u} \cdot \boldsymbol{v} = \sum_{i=1}^{n} \boldsymbol{u}_{[i]} \boldsymbol{v}_{[i]}$$

Geometric

$$\boldsymbol{u} \cdot \boldsymbol{v} = \|\boldsymbol{u}\| \|\boldsymbol{v}\| \cos(\theta)$$

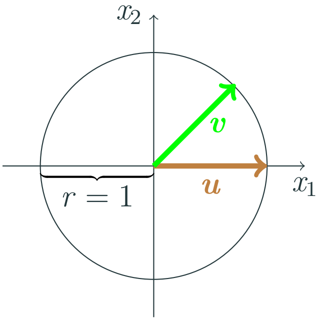


Scalar projection:

$$\Longrightarrow \quad a = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{v}\|}$$

Dot product of unit vectors (aka. cosine similarity)

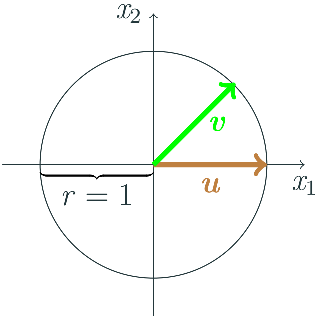
For unit vectors $\boldsymbol{u}, \boldsymbol{v}$:



Dot product of unit vectors (aka. cosine similarity)

For unit vectors $\boldsymbol{u}$, $\boldsymbol{v}$:

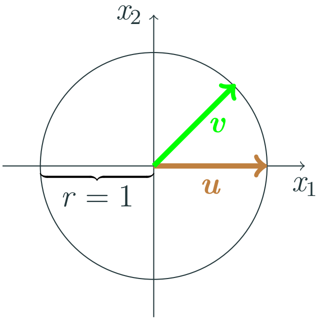
$$\boldsymbol{u} \cdot \boldsymbol{v} = \|\boldsymbol{u}\| \|\boldsymbol{v}\| \cos(\theta) \quad \rightarrow \quad \boldsymbol{u} \cdot \boldsymbol{v} = \cos(\theta)$$

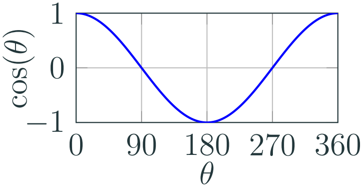


Dot product of unit vectors (aka. cosine similarity)

For unit vectors $\boldsymbol{u}$, $\boldsymbol{v}$:

$$\boldsymbol{u} \cdot \boldsymbol{v} = \|\boldsymbol{u}\|\|\boldsymbol{v}\| \cos(\theta) \quad \rightarrow \quad \boldsymbol{u} \cdot \boldsymbol{v} = \cos(\theta)$$



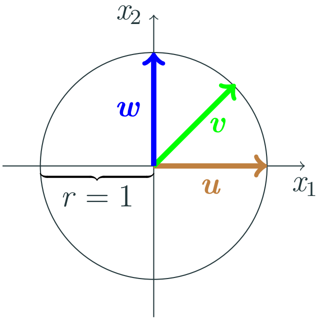


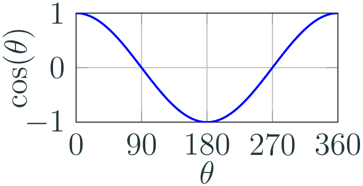
$\boldsymbol{u} \cdot \boldsymbol{v} \approx 0.707$

Dot product of unit vectors (aka. cosine similarity)

For unit vectors $\boldsymbol{u}$, $\boldsymbol{v}$:

$$\boldsymbol{u} \cdot \boldsymbol{v} = \|\boldsymbol{u}\| \|\boldsymbol{v}\| \cos(\theta) \quad \rightarrow \quad \boldsymbol{u} \cdot \boldsymbol{v} = \cos(\theta)$$





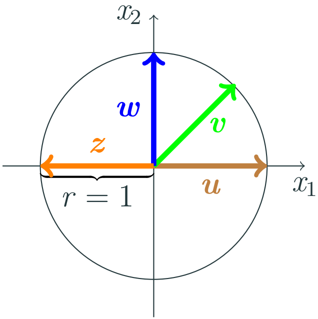
$\boldsymbol{u} \cdot \boldsymbol{v} \approx 0.707$

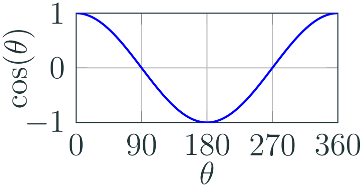
$\boldsymbol{u} \cdot \boldsymbol{w} = 0$ (orthogonal)

Dot product of unit vectors (aka. cosine similarity)

For unit vectors $\boldsymbol{u}$, $\boldsymbol{v}$:

$$\boldsymbol{u} \cdot \boldsymbol{v} = \|\boldsymbol{u}\| \|\boldsymbol{v}\| \cos(\theta) \quad \rightarrow \quad \boldsymbol{u} \cdot \boldsymbol{v} = \cos(\theta)$$



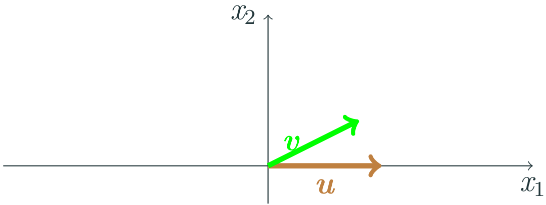


$\boldsymbol{u} \cdot \boldsymbol{v} \approx 0.707$

$\boldsymbol{u} \cdot \boldsymbol{w} = 0$ (orthogonal)

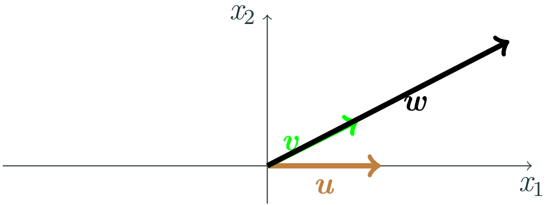
$\boldsymbol{u} \cdot \boldsymbol{z} = -1$ (least 'similar')

Dot product ($\boldsymbol{u} \cdot \boldsymbol{v} = \|\boldsymbol{u}\|\|\boldsymbol{v}\| \cos(\theta)$) is unbounded in $\mathbb{R}$



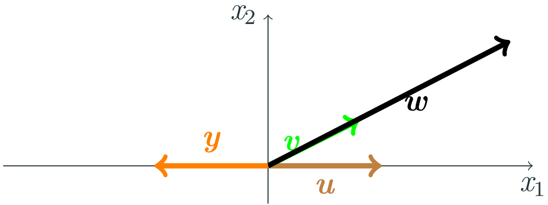
$$\boldsymbol{u} \cdot \boldsymbol{v} > 0$$

Dot product ($\boldsymbol{u} \cdot \boldsymbol{v} = \|\boldsymbol{u}\| \|\boldsymbol{v}\| \cos(\theta)$) is unbounded in $\mathbb{R}$



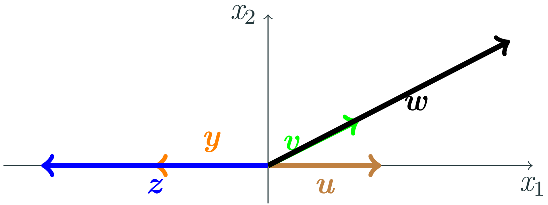
$$\boldsymbol{u} \cdot \boldsymbol{w} > \boldsymbol{u} \cdot \boldsymbol{v} > 0$$

Dot product ($\boldsymbol{u} \cdot \boldsymbol{v} = \|\boldsymbol{u}\| \|\boldsymbol{v}\| \cos(\theta)$) is unbounded in $\mathbb{R}$



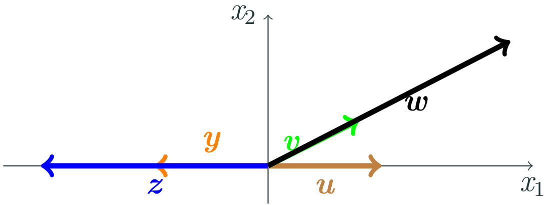
$$\boldsymbol{u} \cdot \boldsymbol{w} > \boldsymbol{u} \cdot \boldsymbol{v} > 0 > \boldsymbol{u} \cdot \boldsymbol{y}$$

Dot product ($\boldsymbol{u} \cdot \boldsymbol{v} = \|\boldsymbol{u}\|\|\boldsymbol{v}\| \cos(\theta)$) is unbounded in $\mathbb{R}$



$$\boldsymbol{u} \cdot \boldsymbol{w} > \boldsymbol{u} \cdot \boldsymbol{v} > 0 > \boldsymbol{u} \cdot \boldsymbol{y} > \boldsymbol{u} \cdot \boldsymbol{z}$$

Dot product ($u \cdot v = \|u\| \|v\| \cos(\theta)$) is unbounded in $\mathbb{R}$



$$\boldsymbol{u} \cdot \boldsymbol{w} > \boldsymbol{u} \cdot \boldsymbol{v} > 0 > \boldsymbol{u} \cdot \boldsymbol{y} > \boldsymbol{u} \cdot \boldsymbol{z}$$

Isn’t it somehow related to Euclidean distance?

Dot product vs. Euclidean distance $\|\boldsymbol{u}-\boldsymbol{v}\|_2 = \sqrt{\sum_{i=1}^{n}\left(\boldsymbol{u}_{[i]}-\boldsymbol{v}_{[i]}\right)^2}$

Let’s take the square of the Euclidean distance

$$\left(\|\boldsymbol{u}-\boldsymbol{v}\|_2\right)^2 = \sum_{i=1}^{n}\left(\boldsymbol{u}_{[i]}-\boldsymbol{v}_{[i]}\right)^2 = \sum_{i=1}^{n}\left(\boldsymbol{u}_{[i]}-\boldsymbol{v}_{[i]}\right)\left(\boldsymbol{u}_{[i]}-\boldsymbol{v}_{[i]}\right)$$

Dot product vs. Euclidean distance $\left\| \boldsymbol{u} - \boldsymbol{v} \right\|_2 = \sqrt{\sum_{i=1}^{n} \left(\boldsymbol{u}_{[i]} - \boldsymbol{v}_{[i]} \right)^2}$

Let's take the square of the Euclidean distance

$$\left(\left\| \boldsymbol{u} - \boldsymbol{v} \right\|_2 \right)^2 = \sum_{i=1}^{n} \left(\boldsymbol{u}_{[i]} - \boldsymbol{v}_{[i]} \right)^2 = \sum_{i=1}^{n} \left(\boldsymbol{u}_{[i]} - \boldsymbol{v}_{[i]} \right) \left(\boldsymbol{u}_{[i]} - \boldsymbol{v}_{[i]} \right)$$

$$= \sum_{i=1}^{n} \left(\boldsymbol{u}_{[i]} \boldsymbol{u}_{[i]} + \boldsymbol{v}_{[i]} \boldsymbol{v}_{[i]} - 2 \boldsymbol{u}_{[i]} \boldsymbol{v}_{[i]} \right)$$

Dot product vs. Euclidean distance $\|\boldsymbol{u} - \boldsymbol{v}\|_2 = \sqrt{\sum_{i=1}^{n} \left(\boldsymbol{u}_{[i]} - \boldsymbol{v}_{[i]}\right)^2}$

Let's take the square of the Euclidean distance

$$\begin{aligned}
\left(\|\boldsymbol{u} - \boldsymbol{v}\|_2\right)^2 &= \sum_{i=1}^{n} \left(\boldsymbol{u}_{[i]} - \boldsymbol{v}_{[i]}\right)^2 = \sum_{i=1}^{n} \left(\boldsymbol{u}_{[i]} - \boldsymbol{v}_{[i]}\right)\left(\boldsymbol{u}_{[i]} - \boldsymbol{v}_{[i]}\right) \\
&= \sum_{i=1}^{n} \left(\boldsymbol{u}_{[i]}\boldsymbol{u}_{[i]} + \boldsymbol{v}_{[i]}\boldsymbol{v}_{[i]} - 2\boldsymbol{u}_{[i]}\boldsymbol{v}_{[i]}\right) \\
&= \sum_{i=1}^{n} \boldsymbol{u}_{[i]}\boldsymbol{u}_{[i]} + \sum_{i=1}^{n} \boldsymbol{v}_{[i]}\boldsymbol{v}_{[i]} - 2\sum_{i=1}^{n} \boldsymbol{u}_{[i]}\boldsymbol{v}_{[i]} \\
&= \boldsymbol{u} \cdot \boldsymbol{u} + \boldsymbol{v} \cdot \boldsymbol{v} - 2\boldsymbol{u} \cdot \boldsymbol{v} \qquad \left(= 2 - 2\boldsymbol{u} \cdot \boldsymbol{v} \text{ for unit vectors}\right)
\end{aligned}$$

Dot product vs. Euclidean distance $\|\boldsymbol{u} - \boldsymbol{v}\|_2 = \sqrt{\sum_{i=1}^{n} \left(\boldsymbol{u}_{[i]} - \boldsymbol{v}_{[i]}\right)^2}$

Let's take the square of the Euclidean distance

$$\begin{aligned}
\left(\|\boldsymbol{u} - \boldsymbol{v}\|_2\right)^2 &= \sum_{i=1}^{n} \left(\boldsymbol{u}_{[i]} - \boldsymbol{v}_{[i]}\right)^2 = \sum_{i=1}^{n} \left(\boldsymbol{u}_{[i]} - \boldsymbol{v}_{[i]}\right)\left(\boldsymbol{u}_{[i]} - \boldsymbol{v}_{[i]}\right) \\
&= \sum_{i=1}^{n} \left(\boldsymbol{u}_{[i]}\boldsymbol{u}_{[i]} + \boldsymbol{v}_{[i]}\boldsymbol{v}_{[i]} - 2\boldsymbol{u}_{[i]}\boldsymbol{v}_{[i]}\right) \\
&= \sum_{i=1}^{n} \boldsymbol{u}_{[i]}\boldsymbol{u}_{[i]} + \sum_{i=1}^{n} \boldsymbol{v}_{[i]}\boldsymbol{v}_{[i]} - 2\sum_{i=1}^{n} \boldsymbol{u}_{[i]}\boldsymbol{v}_{[i]} \\
&= \boldsymbol{u} \cdot \boldsymbol{u} + \boldsymbol{v} \cdot \boldsymbol{v} - 2\boldsymbol{u} \cdot \boldsymbol{v} \qquad \left(= 2 - 2\boldsymbol{u} \cdot \boldsymbol{v} \text{ for unit vectors}\right)
\end{aligned}$$

Conceptual difference: if the origin shifts, the dot product changes, but the distances remains the same

→ Minimizing (square) euclidean distance is proportional to maximizing cosine similarity (equivalent for unit vectors)

Distributional hypothesis

Recap: Neural LMs learn word embeddings

We need to talk about the dot product

Distributional hypothesis

From neural LMs to training word embeddings

word2vec

FastText embeddings: Sub-word embeddings

Advantages and limitations of words embeddings

Recall: One-hot encoding of words

Major drawbacks?

- No ‘semantic’ similarity, all words are equally ‘similar’

Example (see Lecture 3 for more)

$$V = \begin{pmatrix} \text{a}_1 & \text{abandon}_2 & \dots & \text{zone}_{2,999} & \text{zoo}_{3,000} \end{pmatrix}$$

$$\text{nice} = \begin{pmatrix} 0_1 & \dots & 1_{1,852} & \dots & 0_{2,999} & 0_{3,000} \end{pmatrix}$$

$$\text{pleasant} = \begin{pmatrix} 0_1 & \dots & 1_{2,012} & \dots & 0_{2,999} & 0_{3,000} \end{pmatrix}$$

$$\text{horrible} = \begin{pmatrix} 0_1 & \dots & 1_{696} & \dots & 0_{2,999} & 0_{3,000} \end{pmatrix}$$

Distributional hypothesis

The distributional hypothesis stating that *words are similar if they appear in similar contexts*

Example

Intuitively, when we encounter a sentence with an unknown word such as the word **wampinuk** in

Distributional hypothesis

The distributional hypothesis stating that *words are similar if they appear in similar contexts*

Example

Intuitively, when we encounter a sentence with an unknown word such as the word **wampinuk** in

*Marco saw a hairy little **wampinuk** crouching behind a tree*

We infer the meaning of the word based on the context in which it occurs

Count-based embedding methods

Word-context matrices

Long line of research captures the distributional properties of words using word-context matrices

- Each row i represents a word
- Each column j represents a linguistic **context** in which words can occur
- Matrix entry $\boldsymbol{M}_{[i,j]}$ quantifies the **strength of association** between a word and a context

Contexts? Neighboring words, n-grams, etc.

Typically count-based, estimated from a large corpus

Count-based methods

Potential obstacles

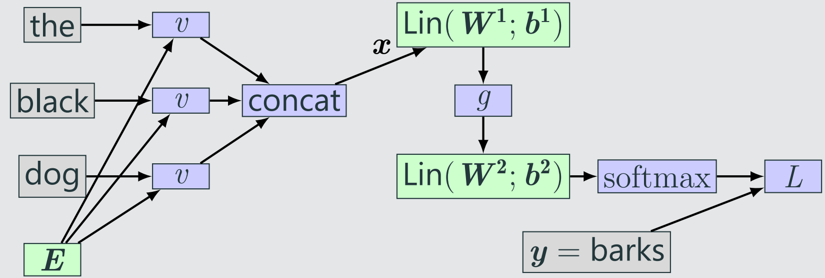
- Data sparsity — some entries in the matrix M may be incorrect because we did not observe enough data points
- The explicit word vectors are of a very high dimensions (depending on the definition of context, the number of possible contexts can be in the hundreds of thousands, or even millions)

$\rightarrow$ Dimensionality reduction techniques, e.g., singular value decomposition (SVD) for low-rank representation (we won’t tackle that)

We have already seen learning representations of words in context

We have already seen learning representations of words in context

Neural LMs



The rows of the embeddings matrix $\boldsymbol{E}$ learn suitable word representation in context (colums of $\boldsymbol{W}^2$ too)

From neural LMs to training word embeddings

Recap: Neural LMs learn word embeddings

We need to talk about the dot product

Distributional hypothesis

From neural LMs to training word embeddings

word2vec

FastText embeddings: Sub-word embeddings

Advantages and limitations of words embeddings

From neural language models to training word embeddings

(Neural) language model's goal: Predict probability distribution over $|V|$ for the next word conditioned on the previous words

- Side product: Can learn useful word embeddings

From neural language models to training word embeddings

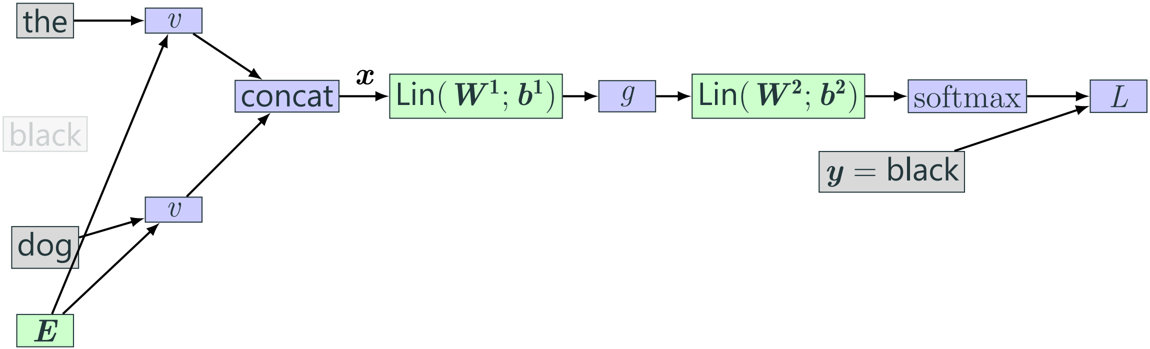
(Neural) language model's goal: Predict probability distribution over $|V|$ for the next word conditioned on the previous words

- Side product: Can learn useful word embeddings

What if we don't need probability distribution but just want to learn word embeddings?

- We can relax our Markov assumption of 'look at k previous words only'
- We can get rid of the costly normalization in softmax

Simplification 1: Ditch the Markov property — look into the future!


the
v
black
$\boldsymbol{x}$
concat
$\mathsf{Lin}(\boldsymbol{W}^1; \boldsymbol{b}^1)$
g
$\mathsf{Lin}(\boldsymbol{W}^2; \boldsymbol{b}^2)$
$\mathrm{softmax}$
L
$\boldsymbol{y} = \mathsf{black}$
v
dog
$\boldsymbol{E}$

For example, insted of modeling $\Pr(w_3|w_1, w_2, \boxdot)$, we model $\Pr(w_2|w_1, \boxdot, w_3)$

Simplification 2: Give up the costly probability distribution

Instead of predicting probability distribution, we just want to predict some score of **context** and **target word**

What could such a score be?

Simplification 2: Give up the costly probability distribution

Instead of predicting probability distribution, we just want to predict some score of **context** and **target word**

What could such a score be?

- Prefer words in their true contexts (high score)

Simplification 2: Give up the costly probability distribution

Instead of predicting probability distribution, we just want to predict some score of **context** and **target word**

What could such a score be?

- Prefer words in their true contexts (high score)
- Penalize words in their ‘untrue’ contexts (low score)

Negative sampling

Instead of predicting probability distribution for the target word, we **create an artificial binary task** by choosing either the correct or a random incorrect target word

Negative sampling

Instead of predicting probability distribution for the target word, we **create an artificial binary task** by choosing either the correct or a random incorrect target word

$$y = \begin{cases} 1 & \text{if } (w, c_{1:k}) \text{ is a positive example from the corpus} \\ 0 & \text{if } (w', c_{1:k}) \text{ is a negative example from the corpus} \end{cases}$$

Negative sampling

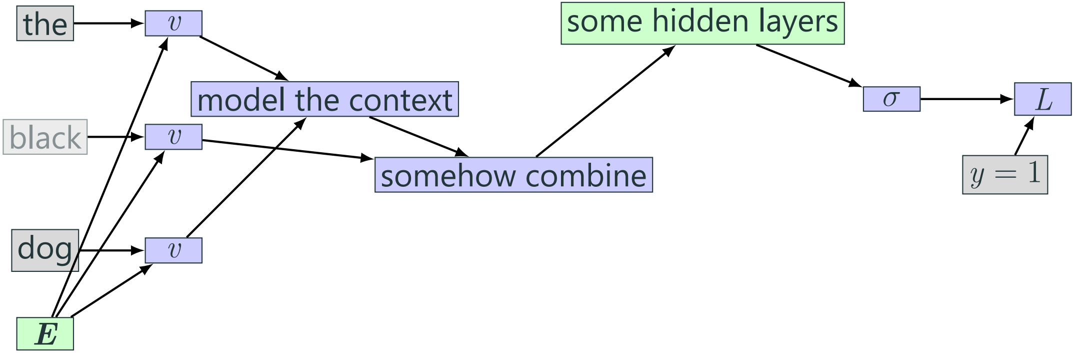
Instead of predicting probability distribution for the target word, we **create an artificial binary task** by choosing either the correct or a random incorrect target word

$$y = \begin{cases} 1 & \text{if } (w, c_{1:k}) \text{ is a positive example from the corpus} \\ 0 & \text{if } (w', c_{1:k}) \text{ is a negative example from the corpus} \end{cases}$$

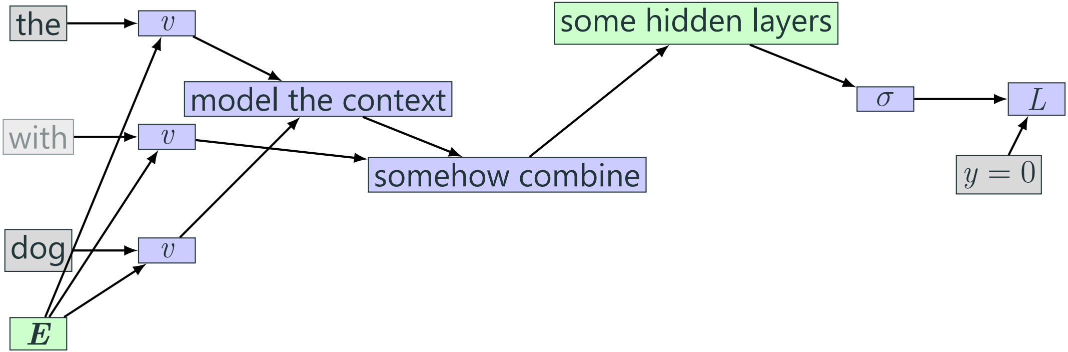
Which distribution for sampling w' from V?

- Corpus-based frequency: $\frac{\#(v)}{\sum_{v' \in V} \#(v')}$ (pr. of each word v)
- Re-weighted: $\frac{\#(v)^{0.75}}{\sum_{v' \in V} \#(v')^{0.75}}$ (more weight on less frequent words done in word2vec)

Turning our problem into a binary classification (positive example)



Turning our problem into a binary classification (negative example)



word2vec

Recap: Neural LMs learn word embeddings

We need to talk about the dot product

Distributional hypothesis

From neural LMs to training word embeddings

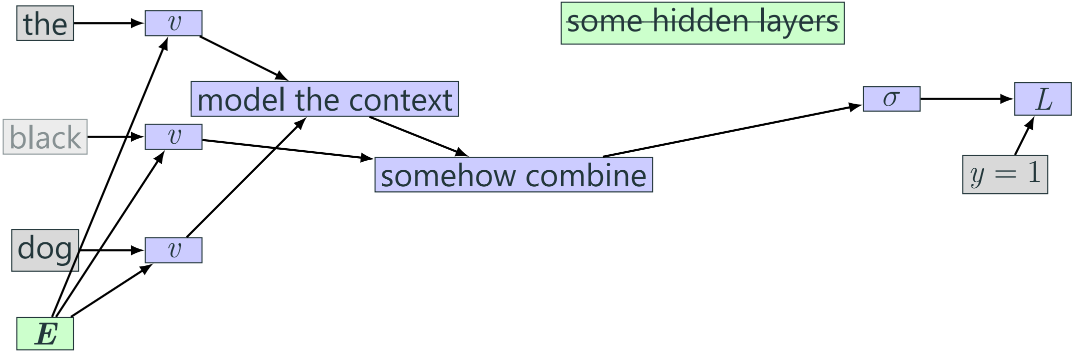
word2vec

FastText embeddings: Sub-word embeddings

Advantages and limitations of words embeddings

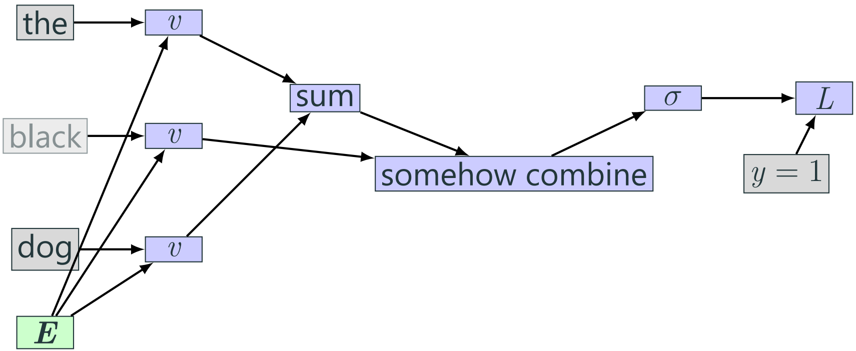
word2vec

word2vec simplifies the neural LM by removing the hidden layer (so turning it into a log-linear model!)



word2vec — how to model the context?

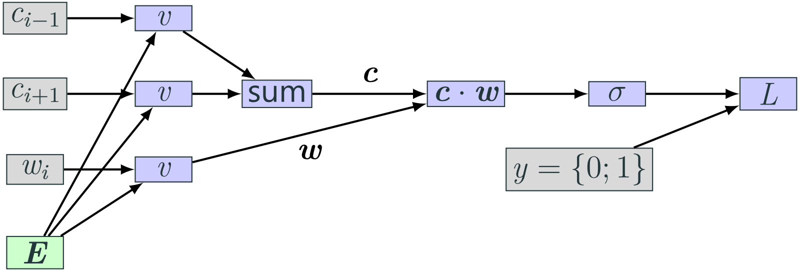
word2vec — how to model the context?



T. Mikolov, K. Chen, G. Corrado, and J. Dean (2013). **"Efficient estimation of word representations in vector space".** In: *1st International Conference on Learning Representations ICLR, Workshop Track Proceedings*. Ed. by Y. Bengio and Y. LeCun, pp. 1–12

Variant 1: Continuous bag of words (CBOW) $\boldsymbol{c} = \sum_{i=1}^{k} v(c_i)$

Final CBOW word2vec — similarity score is the dot product



w_i — target word, c_{i-1}, c_{i+1} — context words (one-hot)

$y = 1$ for correct word-context pairs, $y = 0$ for random w_i

The only learnable parameter is the embedding matrix $\boldsymbol{E}$

What is $\sigma(\boldsymbol{c} \cdot \boldsymbol{w})$ doing?

word2vec: Learning useful word embeddings

Train the network to distinguish ‘good’ word-context pairs from ‘bad’ ones

Create a set D of correct word-context pairs and set $\bar{D}$ of incorrect word-context pairs

The goal of the algorithm is to estimate the probability $\Pr(D = 1|w, c)$ that the word-context pair w, c comes from the correct set D

This should be high (1) for for pairs from D and low (0) for pairs from $\bar{D}$

The corpus-wide loss of word2vec

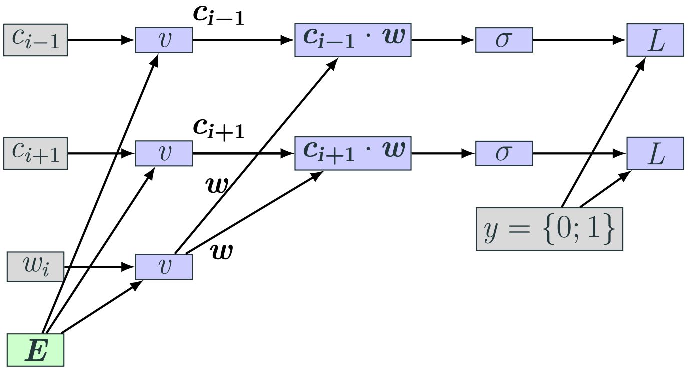
Maximize the log-likelihood of the data $D \cup \bar{D}$

$$\mathcal{L}(\Theta; D, \bar{D}) = \sum_{(w,c) \in D} \log \Pr(D = 1 | w, c) + \sum_{(w',c) \in \bar{D}} \log \Pr(D = 0 | w', c)$$

In word2vec, for each correct word/context, sample k negative pairs into $\bar{D}$, so $\bar{D}$ is k-times larger than D

- k is a hyper-parameter

Skip-Gram — even more relaxed context notion



For k-element context $c_{1:k}$, treat as k different independent contexts (w_i, c_i)

Choosing the context

Sliding window approach — CBOW

Auxiliary tasks are created by taking a sequence of $2m+1$ words

- The middle word is the target (focus) word
- The m words to each side is the context

Sliding window approach — Skip-Gram

$2m$ distinct tasks are created, each pairing the focus word with a different context word

Skip-gram-based approaches shown to be robust and efficient to train

FastText embeddings: Sub-word embeddings

Recap: Neural LMs learn word embeddings

We need to talk about the dot product

Distributional hypothesis

From neural LMs to training word embeddings

word2vec

FastText embeddings: Sub-word embeddings

Advantages and limitations of words embeddings

FastText embeddings

Popular word embedding models ignore the morphology of words, by assigning a distinct vector to each word.

- Limitation, especially for languages with large vocabularies and many rare words

FastText embeddings

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov (2017). **"Enriching Word Vectors with Subword Information".** In: *Transactions of the ACL* 5, pp. 135–146

Popular word embedding models ignore the morphology of words, by assigning a distinct vector to each word.

- Limitation, especially for languages with large vocabularies and many rare words

$\rightarrow$ Model each word as a bag of character n-grams

- Each character n-gram has own embedding
- Word is represented as a sum of n-gram emeddings

FastText embeddings example

Extract all the character n-grams for $3 \leq n \leq 6$

Example

eating $\rightarrow G_w = \{$ <ea, eat, ati, tin, ing, ng>, <eat, eati, atin, ting, ing>, <eati, eatin, ating, ting>, <eatin, eating, ating> $\}$

$$v(\text{eating}) = \sum_{g \in G_w} v(G_w)$$

Train with skip-gram and negative sampling (same as word2vec)

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov (2017). **“Enriching Word Vectors with Subword Information”.** In: *Transactions of the ACL* 5, pp. 135–146

Advantages and limitations of words embeddings

Recap: Neural LMs learn word embeddings

We need to talk about the dot product

Distributional hypothesis

From neural LMs to training word embeddings

word2vec

FastText embeddings: Sub-word embeddings

Advantages and limitations of words embeddings

Using word embeddings

Pre-trained embeddings: ‘Semantic’ input to any neural network instead of one-hot word encoding

- Instance of **transfer learning** — pre-trained (self-trained) on an auxiliary task, plugged into a more complex model as pre-trained weights

Example: Represent a document as an average of its words’ embeddings (average bag-of-words through embeddings) for text classification

Side note: word2vec and word embeddings → part of the deep-learning revolution in NLP around 2015

Semantic similarity, short document similarity, query expansion

S. Kuzi, A. Shtok, and O. Kurland (2016). **"Query Expansion Using Word Embeddings".** In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management.* ACM, pp. 1929–1932

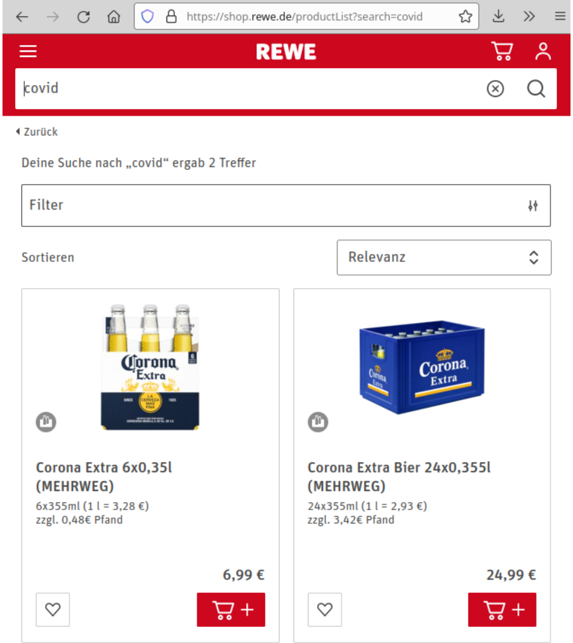
"Using Word2Vec's CBOW embedding approach, applied over the entire corpus on which search is performed, we select terms that are semantically related to the query."

Semantic similarity, short document similarity, query expansion

S. Kuzi, A. Shtok, and O. Kurland (2016). **"Query Expansion Using Word Embeddings".** In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management.* ACM, pp. 1929–1932

"Using Word2Vec's CBOW embedding approach, applied over the entire corpus on which search is performed, we select terms that are semantically related to the query."

What can possibly go wrong?



https://shop.rewe.de/productList?search=covid

REWE

covid

Zurück

Deine Suche nach „covid“ ergab 2 Treffer

Filter

Sortieren

Relevanz

Corona Extra 6x0,35l (MEHRWEG)
6x355ml (1 l = 3,28 €)
zzgl. 0,48€ Pfand

6,99 €

Corona Extra Bier 24x0,355l (MEHRWEG)
24x355ml (1 l = 2,93 €)
zzgl. 3,42€ Pfand

24,99 €

Searched for **covid** (test), returned the closest items with **corona** in the title (because their embeddings learned that covid ≈ corona).

Query expansion with word embeddings might be tricky

Mining word analogies with word2vec

‘Germany to Berlin is France to ?’

Solved by $v(\text{Berlin}) - v(\text{Germany}) + v(\text{France})$, outputs vector $\boldsymbol{x}$ which is closest to Paris in the embeddings space (the closest row in $\boldsymbol{E}$)

Find the queen

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

Interested more in embeddings? Lecture on syntax-based and multilingual word embeddings in DL4NLP 2021 `https://www.youtube.com/watch?v=lnzftxgTAZo`

Limitations of word embeddings (1)

Definition of similarity

Completely operational: words are similar if used in similar contexts

Antonyms

Words opposite of each other (buy—sell, hot—cold) tend to appear in similar contexts (things that can be hot can also be cold, things that are bought are often sold)

Models might tend to judge antonyms as very similar to each other

Limitations of word embeddings (2)

Biases

Distributional methods reflect the usage patterns in the corpora on which they are based

The corpora reflect human biases in the real world (cultural or otherwise)

"Word embeddings encode not only stereotyped biases but also other knowledge [..] are problematic as toward race or gender, or even simply veridical, reflecting the status quo distribution of gender with respect to careers or first names."

A. Caliskan, J. J. Bryson, and A. Narayanan (Apr. 2017). **"Semantics derived automatically from language corpora contain human-like biases".** In: *Science* 356 (6334), pp. 183–186

Limitations of word embeddings (3)

Polysemy, context independent representation

Some words have obvious multiple senses

A *bank* may refer to a financial institution or to the side of a river, a *star* may an abstract shape, a celebrity, an astronomical entity

Using a single vector for all forms is problematic

Recap

Recap: Neural LMs learn word embeddings

We need to talk about the dot product

Distributional hypothesis

From neural LMs to training word embeddings

word2vec

FastText embeddings: Sub-word embeddings

Advantages and limitations of words embeddings

Take aways

- Self-supervised training of word embeddings
- word2vec trained with negative sampling
- CBOW and skip-gram for context modeling

License and credits

Credits

Ivan Habernal

Content from ACL Anthology papers licensed under CC-BY
`https://www.aclweb.org/anthology`