Natural Language Processing with Deep Learning

Lecture 6 — Neural language models and learning word embeddings

Prof. Dr. Ivan Habernal

November 28, 2024

`www.trusthlt.org`
Trustworthy Human Language Technologies Group (TrustHLT)
Ruhr University Bochum & Research Center Trustworthy Data Science and Security

Motivation

I give you a large corpus of 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 ?'

TrustHLT — Prof. Dr. Ivan Habernal   RUHR UNIVERSITÄT BOCHUM   **RU**B

We need to talk about the dot product first

TrustHLT — Prof. Dr. Ivan Habernal

RUHR UNIVERSITÄT BOCHUM RUB

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:

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

TrustHLT — Prof. Dr. Ivan Habernal

RUHR UNIVERSITÄT BOCHUM   **RU**B
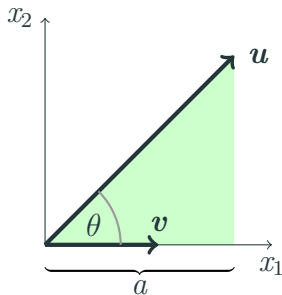
Dot product of unit vectors (aka. cosine similarity)

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

$$\boldsymbol{u} \cdot \boldsymbol{v} = \|\boldsymbol{u}\| \|\boldsymbol{v}\| \cos(\theta) \quad \rightarrow \quad \boldsymbol{u} \cdot \boldsymbol{v} = \cos(\theta)$$
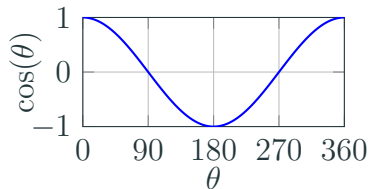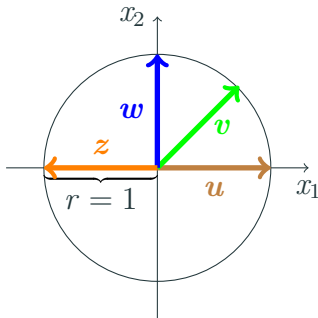
$\boldsymbol{u} \cdot \boldsymbol{v} \approx 0.707$

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

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

TrustHLT — Prof. Dr. Ivan Habernal

RUHR UNIVERSITÄT BOCHUM   RUB

Dot product ($u \cdot v = \|u\| \|v\| \cos(\theta)$) is unbounded in $\mathbb{R}$
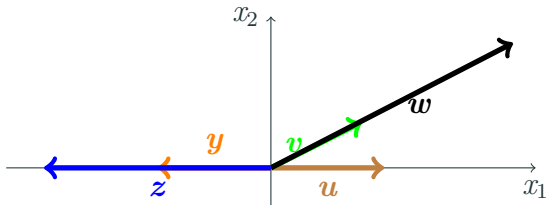
$$u \cdot w > u \cdot v > 0 > u \cdot y > u \cdot z$$

Isn't it somehow related to Euclidean distance?

TrustHLT — Prof. Ivan Habernal

RUHR UNIVERSITÄT BOCHUM   RUB

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

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

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

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

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

$$= \boldsymbol{u} \cdot \boldsymbol{u} + \boldsymbol{v} \cdot \boldsymbol{v} - 2\boldsymbol{u} \cdot \boldsymbol{v} \qquad (= 2 - 2\boldsymbol{u} \cdot \boldsymbol{v} \text{ for unit vectors})$$

$\rightarrow$ Minimizing (square) euclidean distance is *proportional* to maximizing cosine similarity (equivalent for unit vectors)

TrustHLT — Prof. Dr. Ivan Habernal

RUHR UNIVERSITÄT BOCHUM RUB

Language modeling revisited

TrustHLT — Prof. Dr. Ivan Habernal

RUHR
UNIVERSITÄT
BOCHUM

RUB

Recap: What was the task of a language model again?

LM as a conditional probability predictor

Given a sequence of k words, return a probability distribution over V (the vocabulary) for the next possible word

In detail, compute

$$\Pr(W_{k+1} = v \mid W_1, W_2, \ldots, W_k)$$

for all possible words v from vocabulary V

TrustHLT — Prof. Dr. Ivan Habernal

RUHR
UNIVERSITÄT
BOCHUM   RUB

Example

So I was at this party at Joe's place last night, but man I was
so tired, so like at about midnight I said, Joe, I am __

$\Pr(W_{k+1} = v \mid W_1 = \text{'So'}, W_2 = \text{'I'}, \ldots, W_k = \text{'am'})$

Higher $\Pr$:

- leaving
- out
- calling *(it a night)*
- gonna *(head home)*
- . . .

Lower $\Pr$:

- hungry
- in *(love)*
- . . .

TrustHLT — Prof. Dr. Ivan Habernal   RUHR UNIVERSITÄT BOCHUM **RU**B

LM looks like... a text classification task?!

Text classification

Given a text,* predict **its label** from a set of arbitrarily ordered items (**classes such as {sports, politics, travel}**). We model cond. probability for each **label** (a probability distribution over **all the classes**), for example by softmax.

LM

Given a text,* predict **the next word** from a set of arbitrarily ordered items (**the vocabulary**). We model cond. probability for each **word** (a probability distribution over **the entire vocabulary**), for example by softmax.

* We already know how to turn text into a feature vector, e.g., by bag of words

TrustHLT — Prof. Dr. Ivan Habernal

RUHR
UNIVERSITÄT
BOCHUM   RUB

Language modeling revisited

Neural language models

Neural LMs

LM as 'text classification' again

Given a text,* predict **the next word** from a set of
arbitrarily ordered items (**the vocabulary**). We model
cond. probability for each **word** (a probability distribution
over **the entire vocabulary**), for example by softmax.

Let's build a neural network to solve this task

- Input: a k-gram of words $w_{1:k}$
- Desired output: a conditional probability distribution
  over the vocabulary V for the next word w_{k+1}

Embedding layer

If the input are symbolic **categorical features**

- e.g., words from a closed vocabulary

it is common to associate each possible feature value

- i.e., each word in the vocabulary

with a d-dimensional vector for some d

These vectors are also *parameters* of the model, and are trained jointly with the other parameters

TrustHLT — Prof. Dr. Ivan Habernal   RUHR UNIVERSITÄT BOCHUM **RU**B

Embedding layer: Lookup operation

Mapping from a symbolic feature such as
`word-number-48` to d-dimensional vectors is performed
by an embedding layer (a lookup layer)

The parameters in an embedding layer is a matrix $\boldsymbol{E}^{|V| \times d}$,
each row corresponds to a different word in the vocabulary

The lookup operation is then indexing $v(w)$, e.g.,

$$v(w) = v_{48} = \boldsymbol{E}_{[48,:]}$$

If the symbolic feature is a one-hot vector $\boldsymbol{x}$, the lookup
operation can be implemented as the multiplication $\boldsymbol{xE}$

TrustHLT — Prof. Dr. Ivan Habernal   RUHR UNIVERSITÄT BOCHUM   **RU**B

Network concatenating 3 words as embeddings ($d_w = 50$)
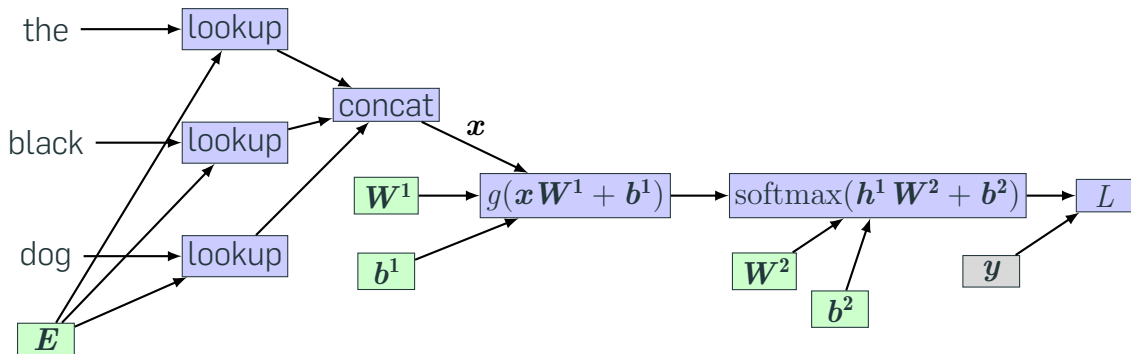
Each word $\in \mathbb{R}^{|V|}$ (one hot), $\boldsymbol{E} \in \mathbb{R}^{|V| \times 50}$, each lookup output
$\in \mathbb{R}^{50}$, concat output $\boldsymbol{x} \in \mathbb{R}^{150}$

TrustHLT — Prof. Dr. Ivan Habernal   RUB

Simplify notation: Lookup as v, linear layer with params
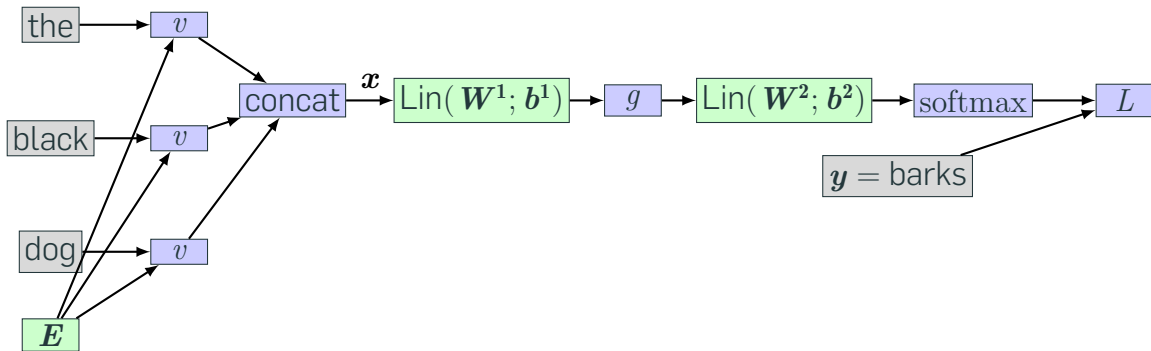
the, black, dog, $\boldsymbol{y}$ = one-hot encoding, $\mathbb{R}^{|V|}$

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

Training neural LMs

Where to get training examples?

Training examples are simply word k-grams from an unlabeled corpus

- Identities of the first $k - 1$ words are used as features
- The last word is used as the target label for the classification

The model is trained using cross-entropy loss

TrustHLT — Prof. Dr. Ivan Habernal   RUHR UNIVERSITÄT BOCHUM   **RU**B

Some advantages of neural LMs

$\approx$ linear increase in parameters with $k + 1$ (better than 'classical' LMs)

TrustHLT — Prof. Dr. Ivan Habernal

RUHR
UNIVERSITÄT
BOCHUM   **RU**B

Learned word representations as a by-product
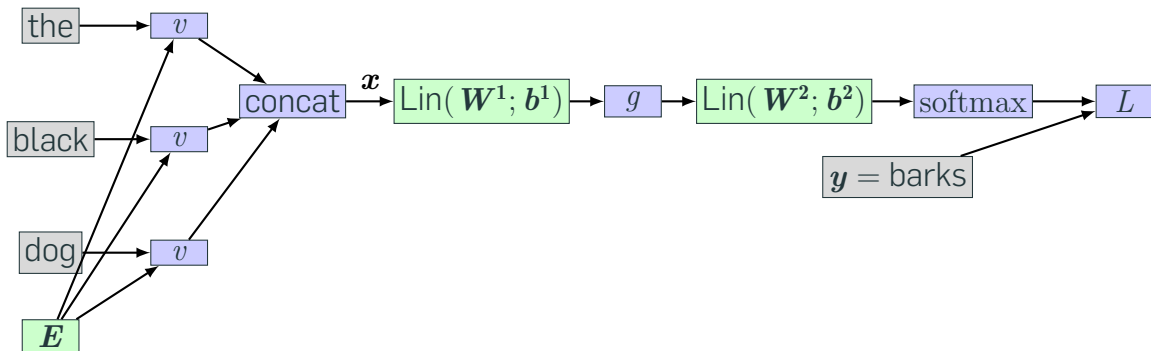
Each row of $\boldsymbol{E}$ learns a word representation

Learning word embeddings

Learning word embeddings

Distributional hypothesis

Recall: One-hot encoding of words

Major drawbacks?

- No 'semantic' similarity, all words are equally 'similar'

Example (see Lecture 3 for more)

$$V = \begin{pmatrix} a_1 & abandon_2 & \dots & zone_{2,999} & zoo_{3,000} \end{pmatrix}$$

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

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

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

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

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

TrustHLT — Prof. Dr. Ivan Habernal

RUHR
UNIVERSITÄT
BOCHUM

RUB

From neural LMs to training 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

TrustHLT — Prof. Dr. Ivan Habernal   RUHR UNIVERSITÄT BOCHUM **RU**B

Simplification 1: Ditch the Markov property — look into the future!
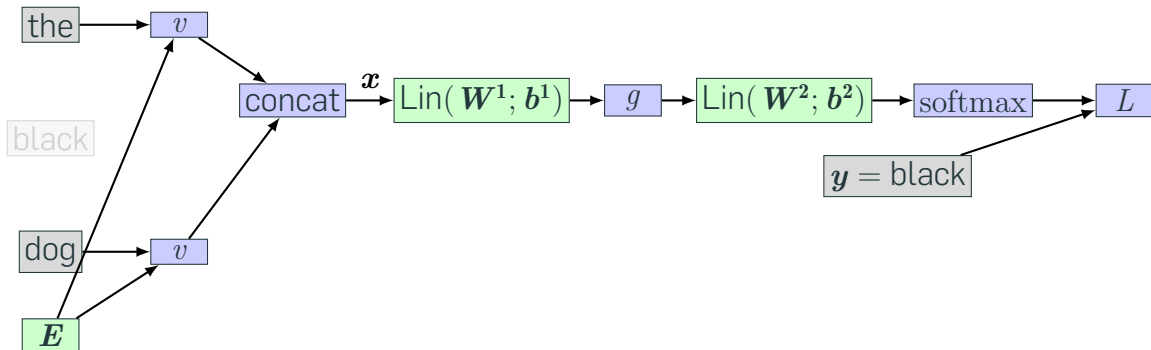
For example, instead of modeling $\Pr(w_3 | w_1, w_2, \boxdot)$, we model
$\Pr(w_2 | w_1, \boxdot, w_3)$

Simplification 2: Give up the costly probability distribution

Instead of predicting probability distribution over the entire vocabulary, 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

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

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)

TrustHLT — Prof. Dr. Ivan Habernal   RUHR UNIVERSITÄT BOCHUM **RUB**

Turn the problem into binary classification (positive example)

TrustHLT — Prof. Dr. Ivan Habernal   RUHR UNIVERSITÄT BOCHUM   **RU**B
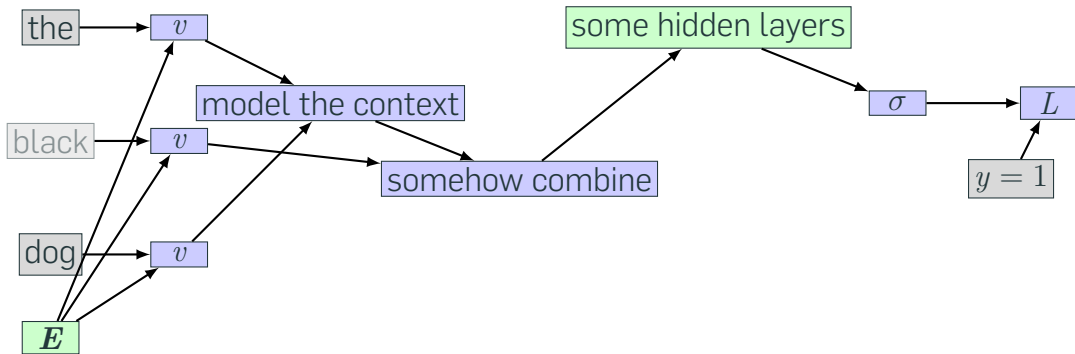
Turn the problem into binary classification (negative example)
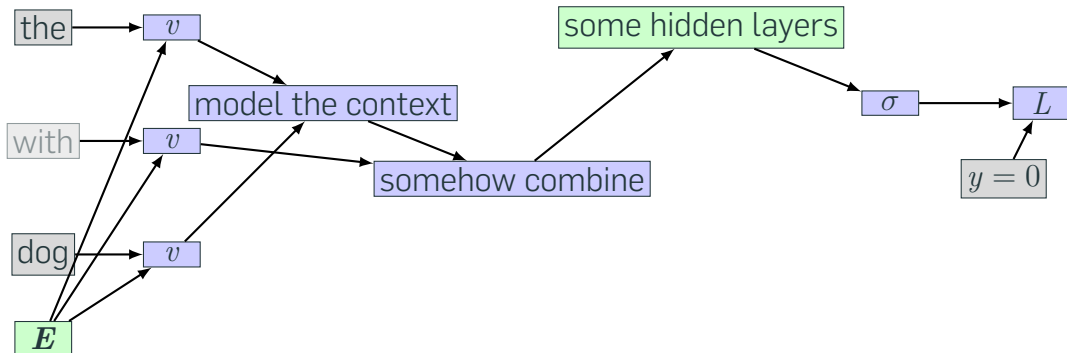
TrustHLT — Prof. Dr. Ivan Habernal   RUB

word2vec

RUHR
UNIVERSITÄT
BOCHUM

RUB

word2vec

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

TrustHLT — Prof. Dr. Ivan Habernal
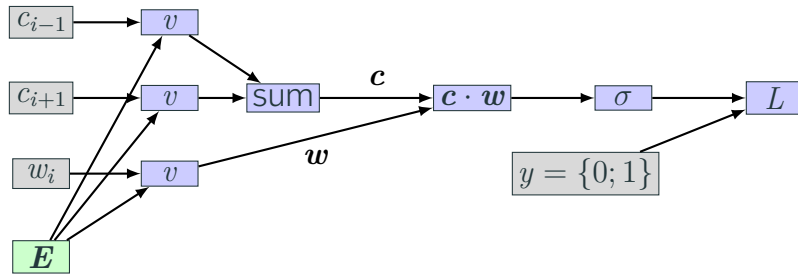
RUHR
UNIVERSITÄT
BOCHUM   **RU**B
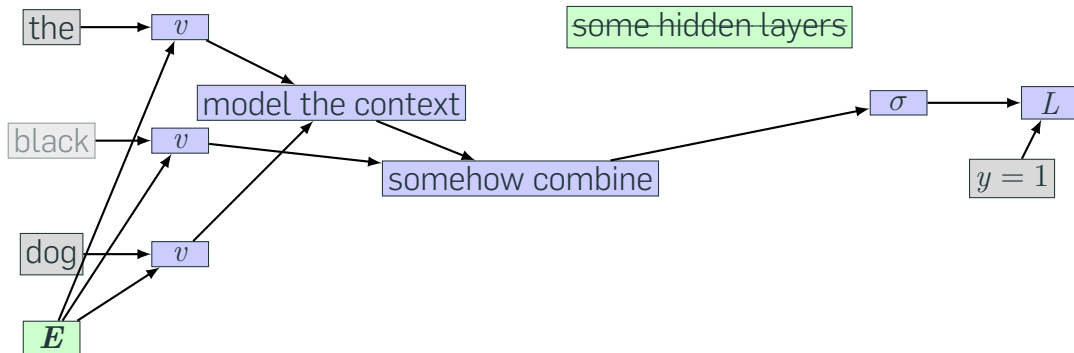
word2vec — how to model the context?
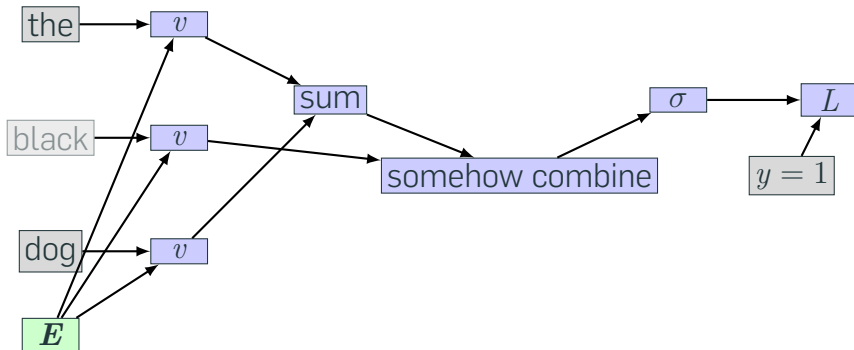
T. Mikolov, K. Chen, G. Corrado, and J. Dean (2013). **"Efficient estimation of word representations in vector space".** In: *1st International Conference on Learning Representations ICLR, Workshop Track Proceedings.* Ed. by Y. Bengio and Y. LeCun, pp. 1–12

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

31   Neural LMs and learning word embeddings   TrustHLT — Prof. Dr. Ivan Habernal   RUHR UNIVERSITÄT BOCHUM   **RUB**

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?

TrustHLT — Prof. Dr. Ivan Habernal

RUHR UNIVERSITÄT BOCHUM **RUB**

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 $\Pr(D = 1 \mid w, c)$ that the word-context pair w, c comes from the correct set D

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

TrustHLT — Prof. Dr. Ivan Habernal   RUHR UNIVERSITÄT BOCHUM **RUB**

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

TrustHLT — Prof. Dr. Ivan Habernal   RUHR UNIVERSITÄT BOCHUM **RU**B

Semantic similarity, short document similarity, query expansion

"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."
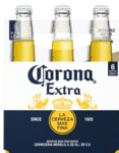
What can possibly go wrong?

TrustHLT — Prof. Dr. Ivan Habernal   RUB

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

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

TrustHLT — Prof. Dr. Ivan Habernal

RUHR
UNIVERSITÄT
BOCHUM   **RU**B

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

TrustHLT — Prof. Dr. Ivan Habernal   RUHR UNIVERSITÄT BOCHUM   **RU**B

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

TrustHLT — Prof. Dr. Ivan Habernal

RUHR UNIVERSITÄT BOCHUM   RUB

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 be an abstract shape, a celebrity, an astronomical entity

Using a single vector for all forms is problematic

Recap

1 We need to talk about the dot product first
2 Language modeling revisited
3 Learning word embeddings
4 From neural LMs to training word embeddings
5 word2vec
6 Advantages and limitations of words embeddings

RUHR
UNIVERSITÄT
BOCHUM RUB

Take aways

- Neural language models
- Self-supervised training of word embeddings
- word2vec trained with negative sampling
- CBOW for context modeling

TrustHLT — Prof. Dr. Ivan Habernal   RUHR UNIVERSITÄT BOCHUM   RUB

License and credits

Licensed under Creative Commons
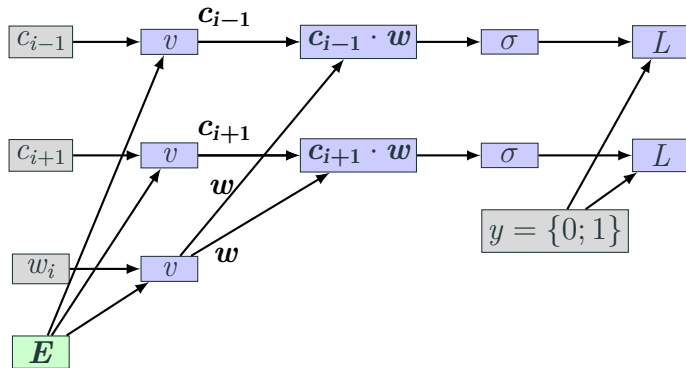Attribution-ShareAlike 4.0 International
(CC BY-SA 4.0)

Credits

Ivan Habernal

TrustHLT — Prof. Dr. Ivan Habernal

Skip-Gram in word2vec

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

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

$\rightarrow$ 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 embeddings

FastText embeddings example

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

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)

Generating text with language models

We can generate ("sample") random sentences from the
model according to their probability

1. Predict a probability distribution over the vocabulary
   conditioned on the start symbol <s>
2. Draw a word from the predicted distribution
3. Predict a probability distribution over the vocabulary
   conditioned on the start symbol and the first generated
   word
4. Draw a word from the predicted distribution
5. Repeat until generated *end-of-sentence* symbol </s>
   (or <EOS>)