

# Continuous Space Representation

PA153

**Pavel Rychlý**

18 Sep 2023

# Problems with statistical NLP

- many distinct words (items) (from Zipf)
- zero counts
  - MLE gives zero probability

$$p(w_3|w_1, w_2) = \frac{\text{count}(w_1, w_2, w_3)}{\text{count}(w_1, w_2)}$$

- not handling similarities
  - some words share some (important) features
  - *driver, teacher, butcher*
  - *small, little, tiny*

# Many distinct words

How to solve:

- use only most frequent ones (ignore outliers)
- use smaller units (subwords)
  - prefixes, suffixes
  - *-er, -less, pre-*

But:

- we want to add more words
- *black hole* is not *black* or *hole*
- even less frequent words are important
  - *deagrofertizace* from “*The deagrofertization of the state must come.*”
  - humans process them easily

## Zero counts

How to solve:

- complicated smoothing strategies
  - Good-Turing, Kneser–Ney, back-off, ...
- bigger corpora
- more data = better estimation

But:

- sometimes there is no more data
  - Shakespeare, new research field
- any size is not big enough

# How big corpus?

Noun *test*

- British National Corpus
- 15789 hits, rank 918
- word sketches from the Sketch Engine
- object-of: *pass, undergo, satisfy, fail, devise, conduct, administer, perform, apply, boycott*
- modifier: *blood, driving, fitness, beta, nuclear, pregnancy*
- can we freely combine any two from that lists?

# How big corpus?

## Collocations of noun *test*

- *blood test* in BNC
  - object-of: *order* (3), *take* (12)
- *blood test* in enClueWeb16 (16 billion tokens)
  - object-of: *order* (708), *perform* (959), *undergo* (174), *administer* (123), *conduct* (229), *require* (676), *repeat* (80), *run* (347), *request* (105), *take* (1215)

# How big corpus?

Phrase *pregnancy test* in 16 billion corpus

**pregnancy test** (*noun*) enClueWeb - Sketches freq = 13677 (0.8 per million)

(test-n filtered by pregnancy)

<u>Constructions</u>	<u>PP_X</u>	<u>955</u>	<u>N_mod</u>	<u>13677</u>	<u>-1.6</u>	<u>and_or</u>	<u>1684</u>	<u>-4.2</u>	
wh	<u>PP in-i</u>	<u>175</u>	-4.8	urine	<u>314</u>	3.07	ultrasound	<u>65</u>	2.25
that_0	<u>PP at-i</u>	<u>150</u>	-3.1	home	<u>2204</u>	2.68	urine	<u>39</u>	1.31
Vinf_to	<u>PP on-i</u>	<u>139</u>	-3.9	blood	<u>248</u>	1.36	counseling	<u>44</u>	0.9
<u>object_of</u>	<u>PP for-i</u>	<u>82</u>	-5.0	serum	<u>53</u>	0.56	condom	<u>23</u>	0.66
take	<u>PP after-i</u>	<u>60</u>	-2.3	at-home	<u>37</u>	0.21	urinalysis	<u>14</u>	0.44
perform	<u>PP with-i</u>	<u>55</u>	-5.1	<u>AVP_post_mod</u>	<u>431</u>	<u>-2.8</u>	test	<u>190</u>	0.33
buy	<u>PP from-i</u>	<u>37</u>	-5.1	prior	<u>27</u>	0.11	smear	<u>14</u>	0.25
administer	<u>PP within-i</u>	<u>32</u>	-3.1	<u>AJ_premod</u>	<u>3077</u>	<u>-3.0</u>	<u>N_premod</u>	<u>1505</u>	nan
	<u>PP to-i</u>	<u>31</u>	-6.6	positive	<u>853</u>	3.66	kit	<u>317</u>	2.48
	<u>PP as-i</u>	<u>26</u>	-5.3				ept	<u>54</u>	1.15
	<u>PP before-i</u>	<u>26</u>	-3.2						

# How big corpus?

Phrase *black hole* in 16 billion corpus

## WORD SKETCH

enTenTen [2012]



black hole 30,327x ...

↔ ⋮ 🔍 ✕	↔ ⋮ 🔍 ✕	↔ ⋮ 🔍 ✕	↔ ⋮
object_of	subject_of	modifier	modifie
accrete ...	accrete ...	supermassive ...	quasar
orbit ...	evaporate ...	super-massive ...	wormhole
gape ...	orbit ...	stellar-mass ...	pulsar
harbor ...	swallow ...	primordial ...	supernova
collide ...	gobble ...	Supermassive ...	quark
evaporate ...	collide ...	intermediate-mass ...	astronomer
harbour ...	devour ...	stellar ...	comet
yawn ...	lurk ...	massive ...	galaxy
rotate ...	coalesce ...	Schwarzschild ...	remnant
		miniature ...	gravity



# Similarities of words

Distinct words?:

- *supermassive, super-massive, Supermassive*
- *small, little, tiny*
- *black hole, star*
- *apple, banana, orange*
- *red, green, orange*
- *auburn, burgundy, mahogany, ruby*

# Continuous space representation

- words are not distinct
- represented by a vector of numbers
- similar words are *closer* each other
- more dimensions = more features
  - tens to hundreds, up to 1000

## Words as vectors

*continue* = [0.286, 0.792, -0.177, -0.107, 0.109, -0.542, 0.349]



# How to create a vector representation

From co-occurrence counts:

- Singular value decomposition (SVD)
  - each word one dimension
  - select/combine important dimensions
  - factorization of co-occurrence matrix
- Principal component analysis (PCA)
- Latent Dirichlet Allocation (LDA)
  - learning probabilities of hidden variables
- Neural Networks

# Neural Networks

- training from examples = supervised training
- sometimes negative examples
- generating examples from texts
- from very simple (one layer) to deep ones (many layers)

# NN training method

- one training example = (input, expected output) = (x, y)
- random initialization of parameters
- for each example:
  - get output for input:  $y' = NN(x)$
  - compute loss = difference between expected output and real output:  
 $loss = |y - y'|$
  - update parameters to decrease loss

# Are vectors better than IDs

- even one hit could provide useful information
- Little Prince corpus (21,000 tokens)
- modifiers of “planet”
  - *seventh, stately, sixth, wrong, tine, fifth, ordinary, next, little, whole*
  - each with 1 hit
  - many are *close* together, share a feature

# Simple vector learning

- each word has two vectors
  - node vector ( $node_w$ )
  - context vector ( $ctx_w$ )
- generate ( $node, context$ ) pairs from text
  - for example from bigrams:  $w_1, w_2$
  - $w_1$  is *context*,  $w_2$  is *node*
- move closer  $ctx_{w_1}$  and  $node_{w_2}$



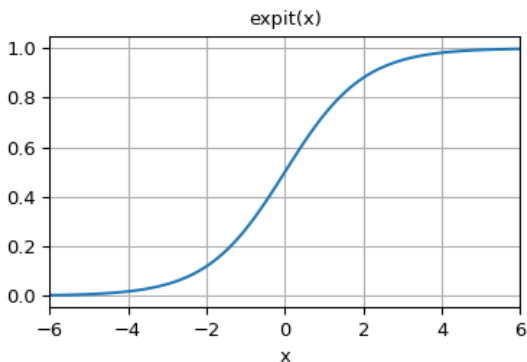
## Simple vector learning

```
node_vec = np.random.rand(len(vocab), dim) * 2 - 1
ctx_vec = np.zeros((len(vocab), dim))
```

```
def train_pair(nodeid, ctxid, alpha):
    global node_vec, ctx_vec
    Nd = node_vec[nodeid]
    Ct = ctx_vec[ctxid]
    loss = 1 - expit(np.dot(Nd, Ct))
    corr = loss * alpha
    Nd += corr * (Ct - Nd)
    Ct += corr * (Nd - Ct)
```

## Expit (sigmoid) function

- $\text{expit}(x) = 1/(1 + \exp(-x)) = 1/(1 + e^{-x})$
- limit range: output in  $(0, 1)$



## Simple vector learning

```
for e in range(epochs):
    last = tokIDs[0]
    for wid in tokIDs[1:]:
        train_pair(wid, last, alpha)
        last = wid
        # update alpha
```

# Embeddings advantages

- no problem in number of parameters
- similarity in many different directions
- good estimations of scores
- **generalization**
  - learnig for some words generalize to similar words

# Embeddings of other items

- lemmata
- part of speech
- topics
- any list of items with some structure

# Summary

- numeric vectors provides continuous space representation of words
- similar words are closer
- similarity in many different directions (features)
  - morphology (number, gender)
  - domain/style
  - word formation