Continuous Space Representation (PA153)

Pavel Rychlý

Problems with statistical NLP

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

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

- 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)

Constructions		
wh	243	-3.6
that_0	212	-4.7
Vinf_to	211	-4.8

object_of	5530	-2.2
take	1765	1.15
perform	203	0.84
buy	237	0.67
administer	40	0.05

PP_X	955	
PP_in-i	175	-4.8
PP_at-i	150	-3.1
PP_on-i	139	-3.9
PP_for-i	82	-5.0
PP_after-i	60	-2.3
PP_with-i	55	-5.1
PP_from-i	37	-5.1
PP_within-i	32	-3.1
PP_to-i	31	-6.6
PP_as-i	26	-5.3
PP_before-i	26	-3.2

N_mod	13677	-1.6
urine	314	3.07
home	2204	2.68
blood	248	1.36
serum	53	0.56
at-home	37	0.21

AVP_post_mod	431	-2.8
prior	27	0.11

AJ_premod	3077	-3.0
positive	853	3.66

and_or	1684	-4.2
ultrasound	65	2.25
urine	39	1.31
counseling	44	0.9
condom	23	0.66
urinalysis	14	0.44
test	190	0.33
smear	14	0.25

N_premod	1505	nan
kit	317	2.48
ept	54	1.15

Figure 1: pregnancy test word sketch

How big corpus?

Phrase *black hole* in 16 billion corpus

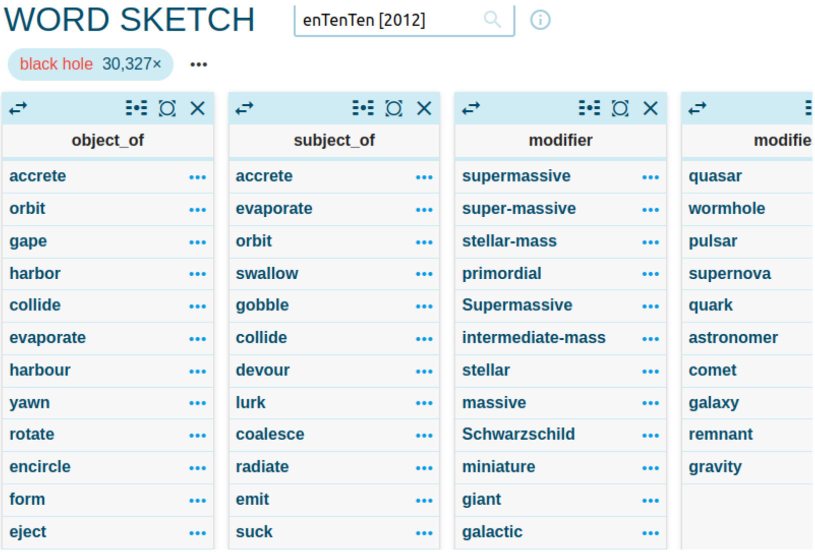

Figure 2: black hole word sketch

Similarities of words

Distinct words?:

- *supermassive, super-massive, Supermassive*
- *small, little, tiny*
- *black hole, star*
- *apple, banana, orange*
- *red, green, orange*
- *auburn, burgundy, mahogony, 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 dimenstions
  - 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 paremeters 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: $w1$, $w2$
  - $w1$ is $context$, $w2$ is $node$
- move closer ctx_{w1} and $node_{w2}$

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]
  corr = (1 - expit(np.dot(Nd, Ct)))* alpha
  Nd += corr * (Ct - Nd)
  Ct += corr * (Nd - Ct)
```

Simple vector learning

```python
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