Deep Learning with Applications in NLP
Course 6

TEXT VECTORIZATION

MIHAELA BREABĂN

1

Agenda

On concepts, senses, similarities

Word vectorizations
- Sparse representations
- Dense representations

Character and document level dense vectorizations

Slides are mainly based on
- Chapter 5 in Dan Jurafsky and James Martin: *Speech and Language Processing (3rd electronic ed., august 2025)* https://web.stanford.edu/~jurafsky/slp3/
- Jeffrey Pennington, Richard Socher, Christopher D. Manning: *GloVe: Global Vectors for Word Representation* https://nlp.stanford.edu/projects/glove/

2

Main question:
How do we represent the meaning of words in NLP systems?

Basically, a word is
- A string of letters -> edit distance -> a topological space -> !semantic
- An index in a vocabulary list -> numerical space -> induces a complete ordering -> !semantic

Better: a vectorial representation based on the index in the vocabulary
- One hot encoding -> Hamming distance -> every two words are at the same distance -> !semantic

Let's start from lexical semantics (the linguistic study of word meaning)

3

Lemmas and senses

lemma

mouse (N)

sense

1. any of numerous small rodents...
2. a hand-operated device that controls a cursor...

Modified from the online thesaurus WordNet

A sense or "concept" is the meaning component of a word
Lemmas can be polysemous (have multiple senses)

4

Relations between words: Synonymy

Synonyms have the same meaning in some or all contexts
- couch / sofa
- big / large
- automobile / car
- vomit / throw up
- water / H_20

5

Relations between words: Synonymy

Note that there are probably no examples of perfect synonymy.
- Even if many aspects of meaning are identical
- Still may differ based on politeness, slang, register, genre, etc.

water/H_20
"H_20" in a surfing guide?
big/large
my big sister != my large sister

In practice, the word *synonym* is therefore used to describe a relationship of approximate or rough synonymy
- Difference in form → difference in meaning

6

Relation: Similarity

Words with similar meanings.

- Not synonyms, but sharing some element of meaning
- While words don't have many synonyms, most words do have lots of *similar* words

```
car, bicycle
cow, horse
cat, dog
```

7

Ask humans how similar 2 words are

https://fh295.github.io/simlex.html

word1	word2	similarity
vanish	disappear	9.8
behave	obey	7.3
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3

SimLex-999 dataset (Hill et al., 2015) - https://arxiv.org/abs/1408.3456v1

8

Relation: Word relatedness
[Budanitsky and Hirst, 2006]

Also called "word association“ in psychology

Words can be related in any way, perhaps via a semantic frame or field

- `coffee, tea`: **similar**
- `coffee, cup`: **related**, not similar
- `scalpel, surgeon`: **related**, not similar

9

Relation: Word relatedness
Semantic field – a common kind of relatedness

Words that
- cover a particular semantic domain
- bear structured relations with each other.

hospitals
surgeon, *scalpel*, *nurse*, *anesthetic*, *hospital*

restaurants
waiter, *menu*, *plate*, *food*, *menu, chef*

houses
door, *roof*, *kitchen*, *family*, *bed*

10

Relation: Antonymy

Senses that are opposites with respect to only one feature of meaning

Otherwise, they are very similar!

`dark/light` `short/long` `fast/slow` `rise/fall`
`hot/cold` `up/down` `in/out`

More formally: antonyms can
- define a binary opposition or be at opposite ends of a scale
  - `long/short, fast/slow`
- Be *reversive*:
  - `rise/fall, up/down`

11

Connotation (sentiment)

- Words have **affective** meanings
  - Positive connotations (*happy*)
  - Negative connotations (*sad*)
- Connotations can be subtle:
  - Positive connotation: *copy, replica, reproduction*
  - Negative connotation: *fake, knockoff, forgery*
- Evaluation (sentiment!)
  - Positive evaluation (*great*, *love*)
  - Negative evaluation (*terrible*, *hate*)

12

Connotation

Words seem to vary along 3 affective dimensions [Osgood et al. (1957)]:

- **valence**: the pleasantness of the stimulus
- **arousal**: the intensity of emotion provoked by the stimulus
- **dominance**: the degree of control exerted by the stimulus

	Word	Score		Word	Score
Valence	love	1.000		toxic	0.008
	happy	1.000		nightmare	0.005
Arousal	elated	0.960		mellow	0.069
	frenzy	0.965		napping	0.046
Dominance	powerful	0.991		weak	0.045
	leadership	0.983		empty	0.081

Values from NRC VAD Lexicon (Mohammad 2018) - https://saifmohammad.com/WebPages/nrc-vad.html

13

So far

Concepts or word senses

- Have a complex many-to-many association with **words** (homonymy, multiple senses)

Have relations with each other

- Synonymy
- Antonymy
- Similarity
- Relatedness
- Connotation

14

Vector semantics

Computational models of word meaning
- the standard way to represent word meaning in NLP
- models many of the aspects of word meaning

The hypothesis:
- "The meaning of a word is its use in the language“ [Ludwig Wittgenstein]
- If A and B have almost identical environments we say that they are synonyms [Zellig Harris, 1954]
- “You shall know a word by the company it keeps” [John Rupert Firth, 1957]

Words are defined by their environments (the words around them)
– distributional hypothesis: words that occur in similar contexts have similar meaning

15

Example: What does recent English borrowing *ongchoi* mean?

Suppose you see these sentences:
- Ong choi is delicious **sautéed with garlic**.
- Ong choi is superb **over rice**
- Ong choi **leaves** with salty sauces

And you've also seen these:
- …spinach **sautéed with garlic over rice**
- Chard stems and **leaves** are **delicious**
- Collard greens and other **salty** leafy greens

Conclusion:
- Ongchoi is a leafy green like spinach, chard, or collard greens
  - We could conclude this based on words like "leaves" and "delicious" and "sauteed"

16

Semantic Vectorial Space

Idea 1:
- Defining meaning by linguistic distribution [Wittgenstein, Harris – 50's]

Idea 2:
- Meaning as a point in multidimensional space [Osgood, 1957]

17

Semantic Vectorial Space

We define meaning of a word as a vector
- Called also "embedding" because it is embedded into a space
- The standard way to represent meaning in NLP

Every modern NLP algorithm uses embeddings as the representation of word meaning

Fine-grained model of meaning for similarity

18

How build word embeddings?

Requires
- A corpus
- An algorithm/method

Embedding types
- Sparse
  - Bag of words, TF-IDF
  - Mutual pointwise information
- Dense
  - Glove
  - Word2vec
  - …

19

Bag of words
TF-IDF

20

Term-document matrix

Each document is represented by a vector of words

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

21

Idea for word meaning: Words can be vectors too!!!

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

battle is "the kind of word that occurs in Julius Caesar and Henry V"
fool is "the kind of word that occurs in comedies, especially Twelfth Night“

What about *the*, *and,…*?

22

TF.IDF

$$w_{t,d} = \log(1 + tf_{t,d}) \times \log\ (N/df_t)$$

Raw counts:

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

tf-idf:

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

23

What is a document?

Could be a play or a Wikipedia article

But for the purposes of tf-idf, documents can be **anything**; we often call each paragraph a document!

24

PMI

More common: word-word co-occurrence matrix (or "term-context matrix")

Two **words** are similar in meaning if their context vectors are similar

is traditionally followed by	**cherry**	pie, a traditional dessert
often mixed, such as	**strawberry**	rhubarb pie. Apple pie
computer peripherals and personal	**digital**	assistants. These devices usually
a computer. This includes	**information**	available on the internet

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Pointwise Mutual Information

Pointwise mutual information:
Do events x and y co-occur more than if they were independent?

$$\text{PMI}(X,Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

PMI between two words: (Church & Hanks 1989)
Do words x and y co-occur more than if they were independent?

$$\text{PMI}(word_1, word_2) = \log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}$$

27

Positive Pointwise Mutual Information

- PMI ranges from $-\infty$ to $+\infty$
- But the negative values are problematic
  - Things are co-occurring **less than** we expect by chance
  - Unreliable without enormous corpora
    - Imagine w1 and w2 whose probability is each 10^{-6}
    - Hard to be sure p(w1,w2) is significantly different than 10^{-12}
- So we just replace negative PMI values by 0
- Positive PMI (**PPMI**) between word1 and word2:

$$\text{PPMI}(word_1, word_2) = \max\left(\log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}, 0\right)$$

28

Computing PPMI on a term-context matrix

Matrix F with W rows (words) and C columns (contexts)

f_{ij} is # of times w_i occurs in context c_j

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \quad p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \quad p_{*j} = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}} \qquad ppmi_{ij} = \begin{cases} pmi_{ij} & \text{if } pmi_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

29

29

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

p(w=information,c=data) = 3982/111716 = .3399

p(w=information) = 7703/11716 = .6575

p(c=data) = 5673/11716 = .4842

	p(w,context)					p(w)
	computer	data	result	pie	sugar	p(w)
cherry	0.0002	0.0007	0.0008	0.0377	0.0021	0.0415
strawberry	0.0000	0.0000	0.0001	0.0051	0.0016	0.0068
digital	0.1425	0.1436	0.0073	0.0004	0.0003	0.2942
information	0.2838	0.3399	0.0323	0.0004	0.0011	0.6575
p(context)	0.4265	0.4842	0.0404	0.0437	0.0052	

30

30

	p(w,context)					p(w)
	computer	data	result	pie	sugar	p(w)
cherry	0.0002	0.0007	0.0008	0.0377	0.0021	0.0415
strawberry	0.0000	0.0000	0.0001	0.0051	0.0016	0.0068
digital	0.1425	0.1436	0.0073	0.0004	0.0003	0.2942
information	0.2838	0.3399	0.0323	0.0004	0.0011	0.6575
p(context)	0.4265	0.4842	0.0404	0.0437	0.0052	

PMI(information,data) = $\log_2$ (.3399 / (.6575*.4842)) = .0944

Resulting PPMI matrix (negatives replaced by 0):

	computer	data	result	pie	sugar
cherry	0	0	0	4.38	3.30
strawberry	0	0	0	4.10	5.51
digital	0.18	0.01	0	0	0
information	0.02	0.09	0.28	0	0

Weighting PMI

PMI is biased toward infrequent events

- Very rare words have very high PMI values

Two solutions:

- Give rare words slightly higher probabilities
- Use add-one smoothing (which has a similar effect)

Weighting PMI: Giving rare context words slightly higher probability

Raise the context probabilities to $\alpha = 0.75$:

$$\text{PPMI}_\alpha(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P_\alpha(c)}, 0)$$

$$P_\alpha(c) = \frac{count(c)^\alpha}{\sum_c count(c)^\alpha}$$

This helps because $P_\alpha(c) > P(c)$ for rare c

Consider two events, P(a) = .99 and P(b)=.01

$$P_\alpha(a) = \frac{.99^{.75}}{.99^{.75}+.01^{.75}} = .97 \quad P_\alpha(b) = \frac{.01^{.75}}{.01^{.75}+.01^{.75}} = .03$$

Bag of words, Tf-idf and PPMI are sparse representations

Increase in size with vocabulary

- length |V|= 20,000 to 50,000
- most elements are zero

Very high dimensional: require a lot of storage

Subsequent classification models have sparsity issues

->Models are less robust

Alternative: dense vectors

vectors which are
- **short** (length 50-1000)
- **dense** (most elements are non-zero)

Short vectors may be easier to use as **features** in machine learning (less weights to tune)

Dense vectors may **generalize** better than storing explicit counts

They may do better at capturing synonymy

In practice, they work better

35

GloVe embeddings

(Pennington et. al, 2014)

GLOBAL VECTORS FOR WORD REPRESENTATION

"While methods like LSA efficiently leverage statistical information, they do relatively poorly on the word analogy task, indicating a sub-optimal vector space structure. Methods like skip-gram may do better on the analogy task, but they poorly utilize the statistics of the corpus since they train on separate local context windows instead of on global co-occurrence counts."

36

Matrix factorization

Related approaches: LSA (topic modeling)
- Term-document matrix
- SVD, NMF

Here: word co-occurence matrices

SVD of the term-term matrix Example

Corpus*: I like deep learning. I like NLP. I enjoy flying*
Context window*: 3*

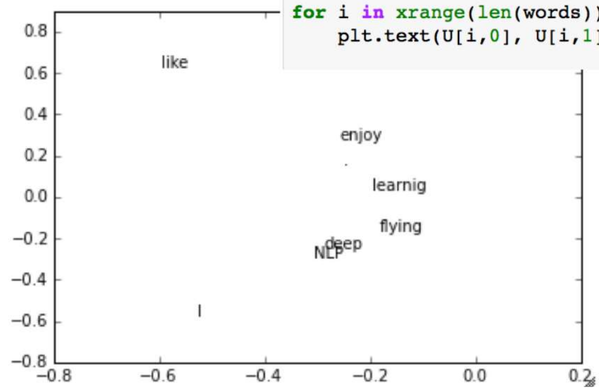
```
import numpy as np
la = np.linalg
words = ["I", "like", "enjoy",
         "deep","learnig","NLP","flying","."]
X = np.array([[0,2,1,0,0,0,0,0],
              [2,0,0,1,0,1,0,0],
              [1,0,0,0,0,0,1,0],
              [0,1,0,0,1,0,0,0],
              [0,0,0,1,0,0,0,1],
              [0,1,0,0,0,0,0,1],
              [0,0,1,0,0,0,0,1],
              [0,0,0,0,1,1,1,0]])

U, s, Vh = la.svd(X, full_matrices=False)
```

SVD word vectors in Python Example

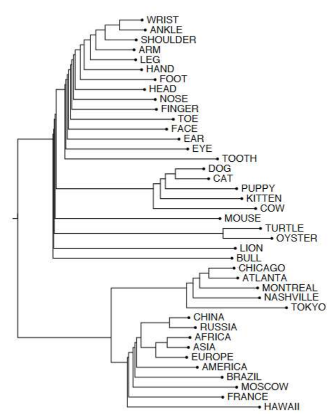
Corpus: I like deep learning. I like NLP. I enjoy flying.
Printing first two columns of U corresponding to the 2 biggest singular values

```
for i in xrange(len(words)):
    plt.text(U[i,0], U[i,1], words[i])
```



39

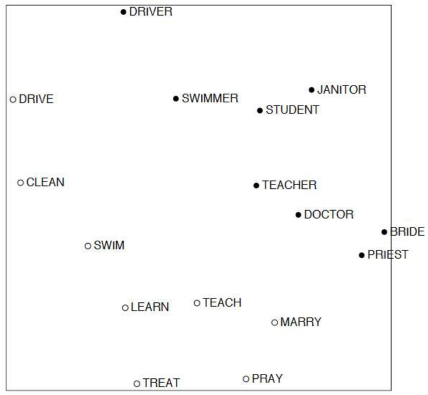
SVD: Interesting semantic patterns



An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence
Rohde et al. 2005

40

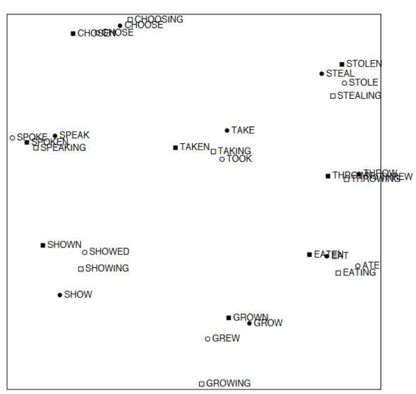
SVD: Interesting semantic patterns



An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence
Rohde et al. 2005

41

SVD: Interesting syntactic patterns



An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence
Rohde et al. 2005

42

Problems with SVD

Computational cost scales quadratically for n x m matrix: O(mn^2) (when n<m)

-> Bad for millions of words

43

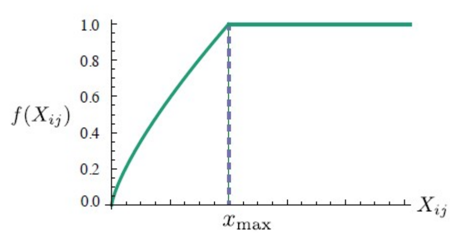
GLOVE: minimize weighted squared loss of the (log)co-occurrence matrix

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

word/context co-occurrence matrix

word vector and bias

context word vector and bias



$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$

Algorithm: stochastic gradient descent
- Fast training
- Scalable to huge corpora

For all our experiments, we set $x_{\max} = 100$, $\alpha = 3/4$, and train the model using AdaGrad (Duchi et al., 2011), stochastically sampling non-zero elements from X, with initial learning rate of 0.05. We run 50 iterations for vectors smaller than 300 dimensions, and 100 iterations otherwise (see

44

How use the two matrices?

For a word we obtain 2 vectors
- As the central word
- As part of the context
- Both capture the co-occurrence information

The set of all vectors generate an approximate decomposition of the log(co-occurrence matrix)

Experiments showed that the best solution is to sum them up

45

Intrinsic word-vector evaluation

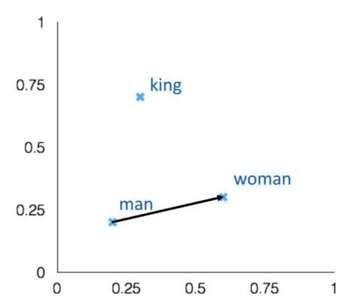
- Word Vector Analogies

a:b :: c:?

man:woman :: king:?

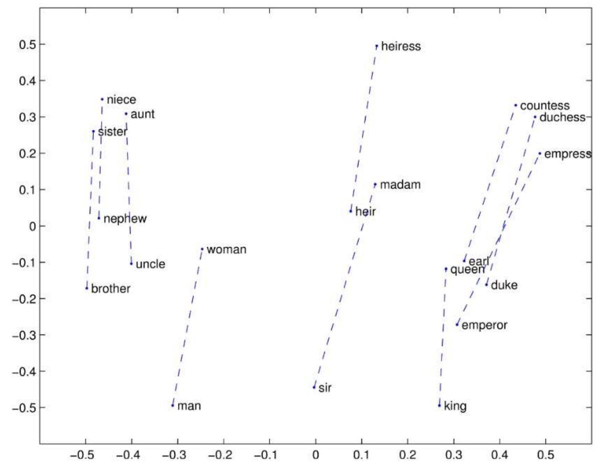
$$d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$$

- Evaluate word vectors by how well their cosine distance after addition captures intuitive semantic and syntactic analogy questions
- Discarding the input words from the search!
- Problem: What if the information is there but not linear?



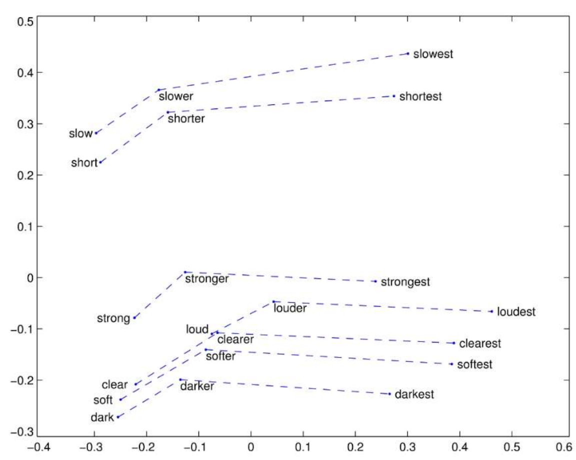
46

Glove visualizations



47

Glove visualizations



48

Word2vec embeddings

(Mikolov et. al, 2013)

Word2vec

Instead of **counting** how often each word *w* occurs near "*apricot*"

- Train a classifier on a **prediction** task:
  - Is *w* likely to show up near "*apricot*"?

We don't actually care about this task

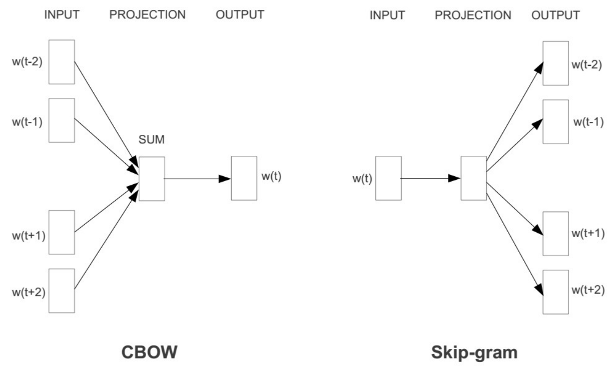
- But we'll take the learned classifier weights as the word embeddings

Big idea: **self-supervision**:

- A word c that occurs near apricot in the corpus acts as the gold "correct answer" for supervised learning
- No need for human labels

Word2vec

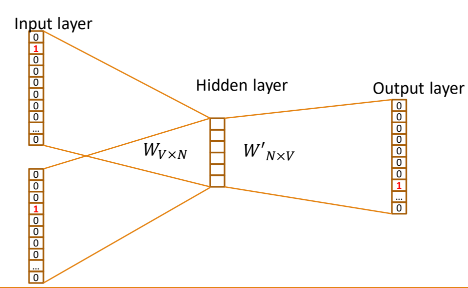
2 basic neural network models:
- **Continuous Bag of Word (CBOW):** use a window of words to predict the middle word
- **Skip-gram (SG):** use a word to predict the surrounding ones in window.



Word2vec
CBOW implementation

Shallow NN:
- Input layer: vocabulary size (input data: context vectors as one hot encoding or their averages)
- 1 hidden layer – ReLU
- Output layer: vocabulary size, SoftMax (multi-class classification)

Cost function: cross-entropy $J = -\sum_{k=1}^{V} y_k \log \hat{y}_k$



Words' representation:
- either W or W'
- or their average

Word2vec
Skip-gram implementation

Similar to CBOW:
- sample (central_word, context_word) as input-output pairs

Issue: SoftMax is computationally expensive

Solution: skip-gram with negative sampling
- Generate positive central_word-context_word pairs by sampling from windows
- Generate negative central_word-context_word pairs by sampling for the central_word a noise context_word from the entire corpus
- -> a binary classification problem

53

Word2vec
Skip-gram implementation

- the probability that word c is a real context word for target word w:

$$P(+|w,c) = \sigma(\mathbf{c}\cdot\mathbf{w}) = \frac{1}{1+\exp(-\mathbf{c}\cdot\mathbf{w})}$$

- the probability that word c is not a real context word for w:

$$P(-|w,c) = 1 - P(+|w,c) = \sigma(-\mathbf{c}\cdot\mathbf{w}) = \frac{1}{1+\exp(\mathbf{c}\cdot\mathbf{w})}$$

- For each context word sample k negative samples and minimize the loss:

$$L = -\log\left[P(+|w,c_{pos})\prod_{i=1}^{k}P(-|w,c_{neg_i})\right] = -\left[\log\sigma(c_{pos}\cdot w)+\sum_{i=1}^{k}\log\sigma(-c_{neg_i}\cdot w)\right]$$

using stochastic gradient descent

54

Negative sampling

Could pick the noise word according to the unigram frequencies P(w)

Better:

$$P_\alpha(w) = \frac{count(w)^\alpha}{\sum_w count(w)^\alpha}$$

α= ¾ works well because it gives rare noise words slightly higher probability

Example:

p(a)=.99 and p(b) = .01:

$$P_\alpha(a) = \frac{.99^{.75}}{.99^{.75} + .01^{.75}} = .97$$

$$P_\alpha(b) = \frac{.01^{.75}}{.99^{.75} + .01^{.75}} = .03$$

55

The kinds of neighbors depend on window size

Small windows (C= +/- 2) : nearest words are syntactically similar words in same taxonomy, same part of speech

Large windows (C= +/- 5) : nearest words are related words in same semantic field

Skip-Gram works well with small datasets, and can better represent less frequent words.

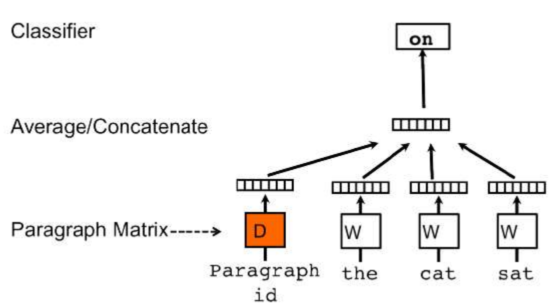
56

Extensions of word2vec
Character and document embeddings

57

Paragraph2vec
(Le, Mikolov, 2014)

Extension of word2vec
Distributed Memory Model of Paragraph Vectors



58

Character and document embeddings

FastText [Bojanowski, Piotr, et al. "Enriching word vectors with subword information." 2017]

- Takes into account morphology
- Word – bag of character n-grams
- Word representation – sum of its n-grams representations

<where>
<wh, whe, her, ere, re>

Sent2vec [Pagliardini et. al"Unsupervised Learning of Sentence Embeddings using Compositional N-Gram Features", 2017]

- Sentence – bag of word n-grams (sub-sentence units)
- Sentence embedding – sum of sub-sentence units

59

Summary

Text vectorization/embedding

- A way to embed the words/text in a numerical multi-dimensional space
- Useful for subsequently applying many ML algorithms

Sparse embeddings vs. dense embeddings

- Capture semantics?

Pre-trained dense word embeddings exist ready for use (extracted from huge corpora), mostly for English but for other languages as well

The presented embeddings are **static**.

Dynamic or contextual embeddings: SentenceTransformers (https://huggingface.co/sentence-transformers)

60

References

Chapter 5 in Dan Jurafsky and James Martin: *Speech and Language Processing (3rdelectronic ed., 2025)*
https://web.stanford.edu/~jurafsky/slp3/

GloVe: https://paperswithcode.com/method/glove

Word2vec: https://arxiv.org/abs/1301.3781

Paragraph2vec: http://proceedings.mlr.press/v32/le14.pdf

fastText: https://github.com/facebookresearch/fastText

https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00051/1567442/tacl_a_00051.pdf

61