Vector semantics

Anton Alekseev,
Steklov Mathematical Institute in St Petersburg NRU ITMO,

NRU ITMO, St Petersburg, 2019
anton.m.alexeyev+itmo@gmail.com

Distributional hypothesis

- Zellig S. Harris: “oculist and eye-doctor... occur in almost the same environments”, “If A and B have almost **identical environments**. . . we say that they are synonyms”

- Most famous, John Firth:
  You shall know a word by the company it keeps!

BTW,
Z. Harris is sometimes referred to as Noam Chomsky’s teacher

John Rupert Firth -- the originator of the London school of linguistics

Harris, Z. S. (1954). Distributional structure. Word, 10, 146–162. Reprinted in J. Fodor and J. Katz, The Structure of Language, Prentice Hall, 1964
Z. S. Harris, Papers in Structural and Transformational Linguistics, Reidel, 1970, 775–794

Firth, J. R. (1957). A synopsis of linguistic theory 1930– 1955. In Studies in Linguistic Analysis. Philological Society. Reprinted in Palmer, F. (ed.) 1968. Selected Papers of J. R. Firth. Longman, Harlow

Words in similar contexts *have similar meaning*

Nothing of **things** that have been **said** was **important**.
Nothing of **stuff** that has been **announced** was **useful**.

I bought X in the nearest shop.
I came home, hung X on the balcony and hung my trousers on it.
The prisoners used X to escape from their cell's window.

Can you guess **what is X**? Any ideas of the properties it has?

What is ‘similarity’?

- **first-order co-occurrence**
  (syntagmatic association)
  Words close in the text, such as:
  ‘drank’ -- and ‘lemonade’/’water’/’tea’

- **second-order co-occurrence**
  (paradigmatic association)
  Words having similar neighbours:
  ‘Tatra’ and ‘Carpathian’, ‘to pet’ and ‘to stroke’

What IS ‘similarity’?

many faces of similarity

- dog -- cat
- dog -- poodle
- dog -- animal
- dog -- bark
- dog -- leash

- dog -- chair — same POS
- dog -- dig — edit distance
- dog -- god — same letters
- dog -- fog — rhyme
- dog -- 6op — shape

https://speakerdeck.com/mlreview/yoav-goldberg-word-embeddings-what-how-and-whither

Every word needs a ‘meaning’ vector

What for?

1. **Most important**: something like transfer learning: instead of BoW
(this way we reuse information from another (possibly bigger) text collection
[and it actually helps])

2. A tool for finding synonyms and other ‘related’ words in some sense

3. Language research tool!
    a. Example: semantic evolution for historians:
    https://nlp.stanford.edu/projects/histwords/
    (there are a few earlier works BTW)

4. **Fun!** quizzes (odd one out), rewriting Great Russian Novels, etc.

Ideas:
how do we learn to find words with similar meaning?

We’ve met before: term-document matrix

But now we care about rows, not columns
(word vectors, not document vectors)

	Zemfira -- Nebomoreoblaka	Sky -- Wikipedia	Fabrika -- The Sea Calls	Eugene Onegin Chapter 1	Anastasia -- The Queen of Gold Sand
sky	6	60		2	
sea	6		10	4	1
cloud	6	18			
love				6	
sand			1		2

We’ve met before: term-document matrix

But now we care about rows, not columns
(word vectors, not document vectors)

	Zemfira -- Nebomoreoblaka	Sky -- Wikipedia	Fabrika -- The Sea Calls	Eugene Onegin Chapter 1	Anastasia -- The Queen of Gold Sand
sky	6	60		2	
sea	6		10	4	1
cloud	6	18			
love				6	
sand			1		2

We'v

But nov

(word v

```
>>> import numpy as np
>>> sea = np.array([6,0,10,4,1])
>>> sand  =np.array([0,0,1,0,2])
>>> cloud = np.array([6,18,0,0,0])

>>> cosine = lambda x,y: x.dot(y) / np.linalg.norm(x) / np.linalg.norm(y)

>>> cosine(sea, sand) > cosine(sea, cloud)
True
>>> cosine(sea, sand) > cosine(sand, cloud)
True
```

sky	6	60		2	
sea	6		10	4	1
cloud	6	18			
love				6	
sand			1		2

Discussion: term-document matrix

- We need A LOT of representative documents, otherwise the approach won’t work

- Dimensionality depends on the text collection size

- Distribution of topics should not be ‘skewed’

- To solve this, maybe we could split documents into subdocuments...
  E.g. sentences? (**NO!** why?)

Discussion: term-document matrix

- We need A LOT of representative documents, otherwise the approach won’t work

- Dimensionality depends on the text collection size

- Distribution of topics should not be ‘skewed’

- To solve this, maybe we could split documents into subdocuments...
  E.g. sentences? (**NO!** why?)

However, looking at smaller **CONTEXT** may be a great idea

Lecture plan

1. ~~Sparse vectors~~
    a. ~~“Term-document” approach~~
    b. “Term-term” approach
        i. Construction
        ii. HAL
    c. Weighting
    d. Semantic similarity estimation
    e. Quality evaluation
2. Dense vectors
    a. Matrix decomposition
    b. “Predictive” approaches

Way better: word-word (word-context) matrix

We count how many times the word occured in the same context with other words (*e.g. in a [-2, 2] window*)

...in Admiralteysky district, St Petersburg. A 37-year old citizen was arrested by [a **police** brigade at around] midnight close to the station of…

...an explosion rambled on Tuesday night close to the entrance of [the **police** station in the] city of Helsingborg...

...the unknown with cold steel arms attacked [the police brigade at the] gas station...

In [Vyborg, police station might eventually] catch fire...

We get sparse vectors with a large number of dimensions

contexts →

words ↓

	brigade	city	police	building
brigade	x	...	...	...
city	...	x	...	...
police	2	1	x	2
building	...	...	...	...
..				
militia	3	0	1	4

Similar words have almost the same row cells filled

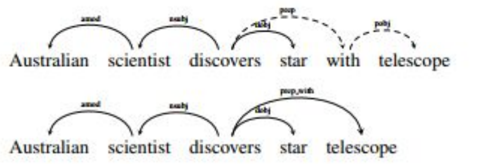
Way better: word-word (word-context) matrix

Important: there are many ways to **define ‘co-occurence’**

E.g., one can choose a ‘syntactically motivated’ part of a sentence as a context -- instead of a window

see. "**Dependency-Based Word Embeddings**", Omer Levy and Yoav Goldberg, 2014

(however, this paper is on dense vectors, the ones we haven’t yet discussed)



WORD	CONTEXTS
australian	scientist/amod^{-1}
scientist	australian/amod, discovers/nsubj^{-1}
discovers	scientist/nsubj, star/dobj, telescope/prep_with
star	discovers/dobj^{-1}
telescope	discovers/prep_with^{-1}

The choice of context window defines vector’s properties

1. Small window -- ~ ‘syntactic’ similarity
2. Larger window -- ~ ‘meaning’ similarity

Lecture plan

1. ~~Sparse vectors~~
    a. ~~“Term-document” approach~~
    b. ~~“Term-term” approach~~
        i. ~~Construction~~
        ii. HAL
    c. Weighting
    d. Semantic similarity estimation
    e. Quality evaluation
2. Dense vectors
    a. Matrix decomposition
    b. “Predictive” approaches

Example: HAL (Hyperspace Analogue to Language)

Oldschool example: ‘window approach’ where we increment counters for ALL pairs of words in a window

This way the words that are closer to each other in the window get more ‘weight’

Table 1
Example Matrix for “The Horse Raced Past the Barn Fell”
(Computed for Window Width of Five Words)

	barn	fell	horse	past	raced	the
<PERIOD>	4	5	0	2	1	3
barn	0	0	2	4	3	6
fell	5	0	1	3	2	4
horse	0	0	0	0	0	5
past	0	0	4	0	5	3
raced	0	0	5	0	0	4
the	0	0	3	5	4	2

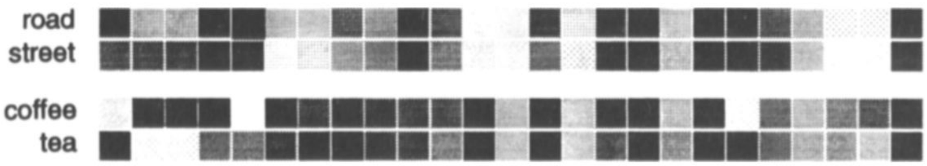


Figure 1. Gray-scaled 25-element co-occurrence vectors.

Lund, K., Burgess, C. & Atchley, R. A. (1995). Semantic and associative priming in a high-dimensional semantic space. Cognitive Science Proceedings (LEA), 660-665.

Lund, K., & Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. Behavior Research Methods, Instrumentation, and Computers, 28, 203-208.

Example: HAL (Hyperspace Analogue to Language)

Table 2
Five Nearest Neighbors for Target Words From Experiment 1 (*n*1 . . . *n*5)

Target	*n*1	*n*2	*n*3	*n*4	*n*5
jugs	juice	butter	vinegar	bottles	cans
leningrad	rome	iran	dresden	azerbaijan	tibet
lipstick	lace	pink	cream	purple	soft
triumph	beauty	prime	grand	former	rolling
cardboard	plastic	rubber	glass	thin	tiny
monopoly	threat	huge	moral	gun	large

Lund, K., Burgess, C. & Atchley, R. A. (1995). Semantic and associative priming in a high-dimensional semantic space. Cognitive Science Proceedings (LEA), 660-665.

Lund, K., & Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. Behavior Research Methods, Instrumentation, and Computers, 28, 203-208.

Disadvantages of ‘simple counts’

Counts assign large values to ‘useless’ words (in terms of meaning) such as prepositions, articles, etc. However they do not add any useful information.

Question: any ideas on how to modify **weight(word, context),** so that useless yet frequent words won’t have large weight?

Let’s use ‘importances’ as weights

We know at least two ways to do it

For term-document vectors (discussed earlier):

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$$

...simple **idf** is also valid.

For term-term case:

$$\text{pmi}(x; y) \equiv \log \frac{p(x, y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)}.$$

Also: be careful when removing stop-words!

PMI-weighted word-context matrix

Estimating probabilities as frequencies of occurrences within the same window for a given word

contexts

$$\mathrm{PMI}(w,c) = \log_2 \frac{P(w,c)}{P(w)P(c)}$$

	brigade	city	police	building
brigade	x	...	...	...
city	...	x	...	...
police	2	1	x	2
building	...	...	...	...
..				
militia	3	0	1	4

words

p(w) = count(police, *) / all =
sum of ‘police’ row / sum of matrix elements

p(c) = count(*, station) / all =
sum of ‘station’ row / sum of matrix elements

p(w, c) = count(police station) / all =
2 / sum of matrix elements

Positive PMI (PPMI)

We often have to deal with rare words (e.g. one in a million), thus checking whether two events with probabilities lower than 10^-6 (estimated as a simple fraction of counts) is a bad idea :(

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

$$\text{PPMI}_{ij} = \max(\log_2 \frac{p_{ij}}{p_{i*} p_{*j}}, 0)$$

Problem: (P)PMI "likes" rare events

Omer Levy, Yoav Goldberg, Ido Dagan introduced a trick to deal with it in 2015:

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

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

...Inspired by similar ideas in word2vec and GloVe implementations

A value of 0.75 showed the best performance on all tasks

(though may need tuning on your task!)

Levy, O., Goldberg, Y., and Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. TACL, 3, 211–225.

Other weighting schemes

Student’s t-test: estimation how far from each other are observed mean and expected mean

“Can we reject this hypothesis?”

$$t = \frac{\bar{x} - \mu}{\sqrt{\frac{s^2}{N}}}$$

$$P(a,b) = P(a)P(b)$$

$$\text{t-test}(a,b) = \frac{P(a,b) - P(a)P(b)}{\sqrt{P(a)P(b)}}$$

One can use this statistic for collocations extraction as well

Почему так можно?

Manning, C. D. and Schutze, H. (1999). ¨ Foundations of Statistical Natural Language Processing. MIT Press.
Curran, J. R. (2003). From Distributional to Semantic Similarity. PhD thesis

Lecture plan

1. ~~Sparse vectors~~
    a. ~~"Term-document" approach~~
    b. ~~"Term-term" approach~~
        i. ~~Construction~~
        ii. ~~HAL~~
    c. ~~Weighting~~
    d. Semantic similarity estimation
    e. Quality evaluation
2. Dense vectors
    a. Matrix decomposition
    b. "Predictive" approaches

Vector closeness estimation

We already know one way to do it

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}}$$

Another view on this:

1. scalar product is a ‘weighted set intersection cardinality’
2. we need the denominator as a way to heal scalar product’s tendency to grow because of the large vector values (possibly few)

Vector closeness estimation - 2

“Soft” Jaccard distance (context = set element)

$$\text{sim}_{\text{Jaccard}}(\vec{v},\vec{w}) = \frac{\sum_{i=1}^{N} \min(v_i, w_i)}{\sum_{i=1}^{N} \max(v_i, w_i)}$$

Normalize vectors so that the sum of values of each equals to 1 and compute the KL-divergence between them

$$D(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

Is that OK?

Vector closeness estimation - 2

“Soft” Jaccard distance (context = set element)

$$\text{sim}_{\text{Jaccard}}(\vec{v},\vec{w}) = \frac{\sum_{i=1}^{N}\min(v_i,w_i)}{\sum_{i=1}^{N}\max(v_i,w_i)}$$

Normalize vectors so that the sum of values of each equals to 1 and compute the KL-divergence between them

$$D(P||Q) = \sum_x P(x)\log\frac{P(x)}{Q(x)}$$

We may have zeros we can’t divide by **or take logarithm of**

Vector closeness estimation* - 3

Symmetric distance based on Kullback-Leibler divergence:

$$D(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

[Jensen-Shannon divergence](), a sum of KL-d between each distribution and an average distribution

$$JS(P||Q) = D(P|\frac{P+Q}{2}) + D(Q|\frac{P+Q}{2})$$

In our case it looks like this

$$\text{sim}_{\text{JS}}(\vec{v}||\vec{w}) = D(\vec{v}|\frac{\vec{v}+\vec{w}}{2}) + D(\vec{w}|\frac{\vec{v}+\vec{w}}{2})$$

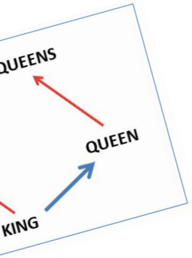
Lecture plan

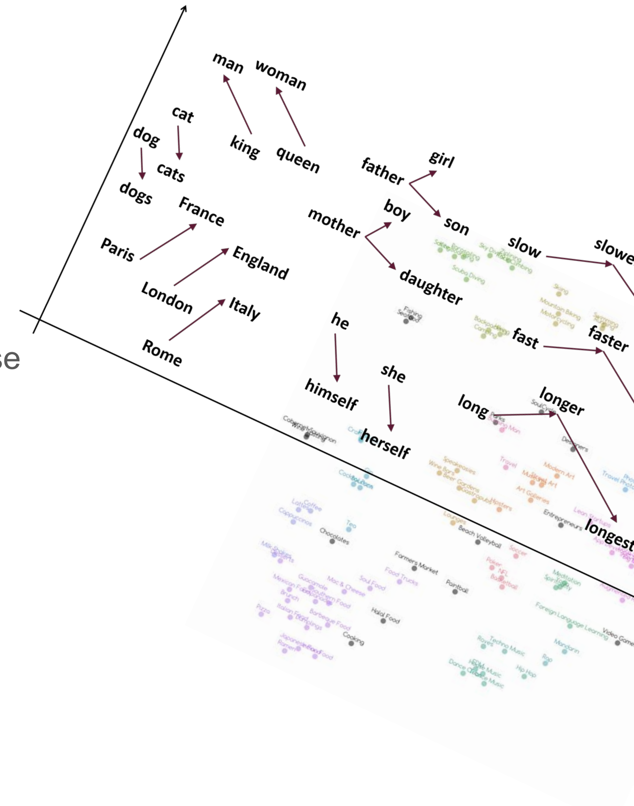
1. ~~Sparse vectors~~
    a. ~~“Term-document” approach~~
    b. ~~“Term-term” approach~~
        i. ~~Construction~~
        ii. ~~HAL~~
    c. ~~Weighting~~
    d. ~~Semantic similarity estimation~~
    e. Quality evaluation
2. Dense vectors
    a. Matrix decomposition
    b. “Predictive” approaches

Word vectors quality evaluation

1. **Extrinsic evaluation**
   the best way to estimate word vectors quality for practical tasks. E.g.:
   a. short texts classification
   b. any other useful task :)

2. **Intrinsic evaluation**
   a. mainstream: evaluation on pairs of words that are ‘similar’ in some sense
   b. mainstream: syntactic/semantic analogy tasks
   c. clustering words labeled with ‘groups’ (+computing purity)
   d. ...a few more ideas





See also: Schnabel, Tobias & Labutov, Igor & Mimno, David & Joachims, Thorsten. (2015). Evaluation methods for unsupervised word embeddings. 298-307. 10.18653/v1/D15-1036.
Don’t count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors M Baroni, G Dinu, G Kruszewski Proceedings of Association for Computational Linguistics (ACL) 1

Lecture plan

1. ~~Sparse vectors~~
    a. ~~“Term-document” approach~~
    b. ~~“Term-term” approach~~
        i. ~~Construction~~
        ii. ~~HAL~~
    c. ~~Weighting~~
    d. ~~Semantic similarity estimation~~
    e. ~~Quality evaluation~~
2. Dense vectors
    a. Matrix decomposition
    b. “Predictive” approaches

Reminder

We already know sparse representations:
term-term/term-document counts/weights

1) how to build the matrix
2) a few ways to set weights
3) tricks to tune
4) how to evaluate (extrinsic/intrinsic)

“Dense” vectors

- tens of thousands dimensions to hundreds dimensions
- small number of zeros
- moving away from approach ‘coordinate=term’

But... why would we do it?

Sparse vectors we’ve discussed assign every word a coordinate, hence

- models using sparse vectors as input are hard to train: a large number of parameters sometimes makes machine learning models too complex

- it is hard to ‘grasp’ synonymy as contexts-synonyms simply have different and unrelated coordinates

Main approaches

1. Matrix factorization
2. “Predictive”, “neural” approaches
3. Word clustering

Lecture plan

1. ~~Sparse vectors~~
    a. ~~“Term-document” approach~~
    b. ~~“Term-term” approach~~
        i. ~~Construction~~
        ii. ~~HAL~~
    c. ~~Weighting~~
    d. ~~Semantic similarity estimation~~
    e. ~~Quality evaluation~~
2. Dense vectors
    a. Matrix decomposition
    b. “Predictive” approaches

Matrix decomposition

Intuition:

1) we decrease the number of dimensions hoping to keep the regularities and laws present in the data (e.g., synonymy),

2) one may want to keep only the most ‘important’ coordinates (the ones that have the largest variance in values)

SVD: singular value decomposition

Any matrix can be represented like this

$$A = USV^T$$

where **S** is **a diagonal matrix** (having the same dimensions as A), values on diagonals are singular values, **U, V are orthogonal**

Eckart-Yang theorem
the best possible **rank k approximation of the matrix A** (in terms of Frobenius norm) is a singular value decomposition, where in the resulting matrix **S** only first **k diagonal elements** are non-zero and are ordered in non-increasing order.

Lower rank approximation

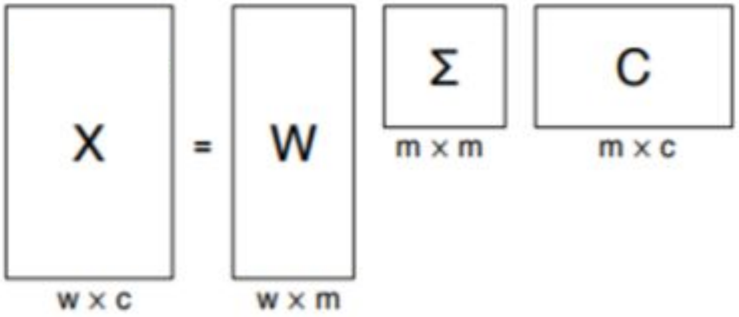
The task can be posed in a different way

W: matrix: **w words** x **m dimensions** of the 'latent space', and

- columns are orthogonal to each other
- columns are ordered in the order of decreasing variance in coordinates in a new space

Σ: diagonal matrix **m** x **m**, where each value on the diagonal reflects the 'importance' of the corresponding dimension

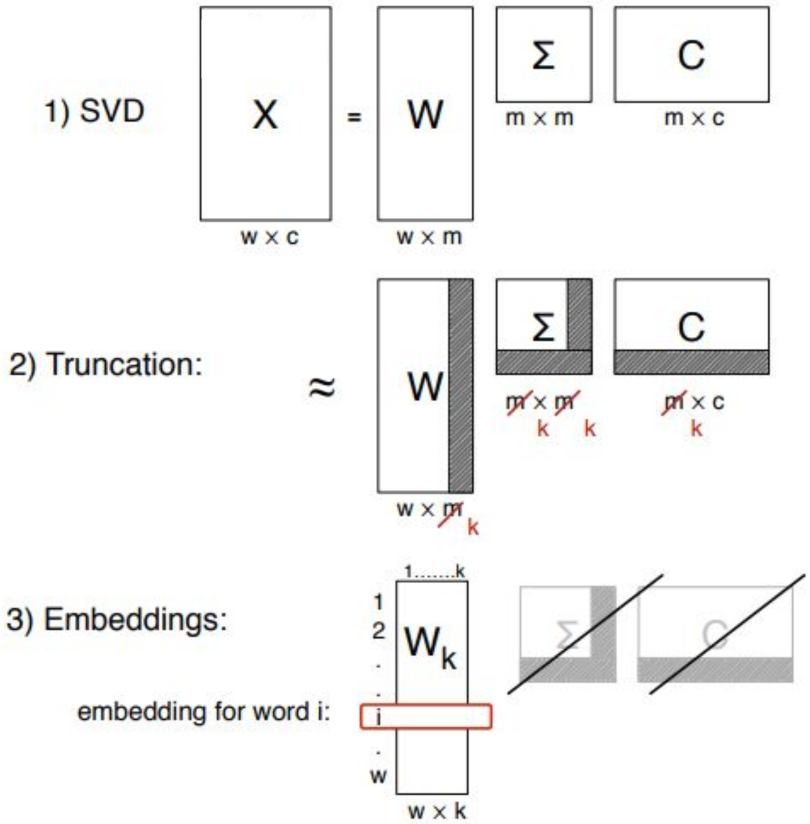
C: matrix: **m** x **c**



Truncated SVD

Letting only top K dimensions live

Then our word vector representations are corresponding rows in matrix W_k , that is, k-dimensional vectors



LSA: Latent Semantic Analysis

	access	document	retrieval	information	theory	database	indexing	computer
Doc 1	x	x	x			x	x	
Doc 2				x*	x			x*
Doc 3			x	x*				x*

Applying SVD (**m** = hundreds) to term-document matrix, setting weights as a product of:

the local weight

$$\log f(i,j) + 1$$

the global weight

$$1 + \frac{\sum_j p(i,j) \log p(i,j)}{\log D}$$

for all terms **i** in all documents **j**

S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman. 1988. Using latent semantic analysis to improve access to textual information. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '88), J. J. O'Hare (Ed.). ACM, New York, NY, USA, 281-285.

Truncated SVD for term-term PPMI matrix

We simply apply SVD to word-context matrix and cut off some of the dimensions, choosing **k** manually. Sometimes works better than the sparse analogue.

Other notes on SVD as a way of obtaining vector representations of words:

- $(W\Sigma)^T$ can also be treated and used as word vectors (it doesn't work, though)
- Truncating (you never know, but it seems so) helps to generalize and filter out useless information,
- Sometimes throwing away the **first few dimensions** may be helpful

However, it is computationally hard

Lecture plan

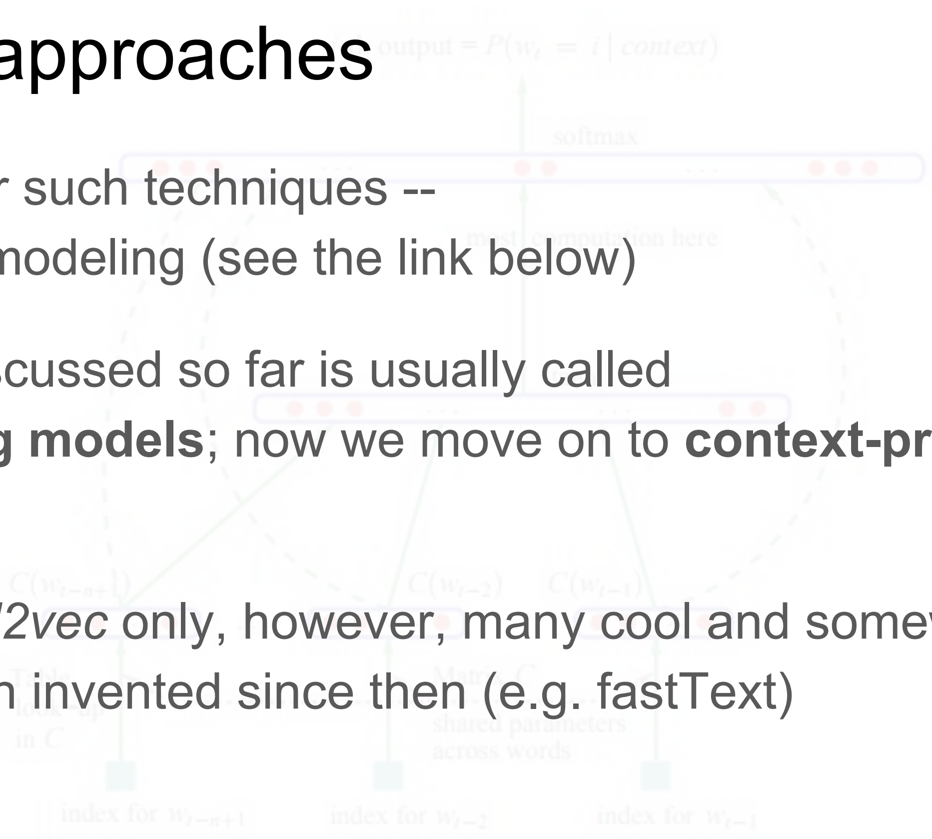
1. ~~Sparse vectors~~
    a. ~~“Term-document” approach~~
    b. ~~“Term-term” approach~~
        i. ~~Construction~~
        ii. ~~HAL~~
    c. ~~Weighting~~
    d. ~~Semantic similarity estimation~~
    e. ~~Quality evaluation~~
2. ~~Dense vectors~~
    a. ~~Matrix decomposition~~
    b. “Predictive” approaches

‘Predictive’ approaches

The inspiration for such techniques --
neural language modeling (see the link below)

What we have discussed so far is usually called **context-counting models**; now we move on to **context-predicting models**

We’ll look at *word2vec* only, however, many cool and somewhat similar models have been invented since then (e.g. fastText)



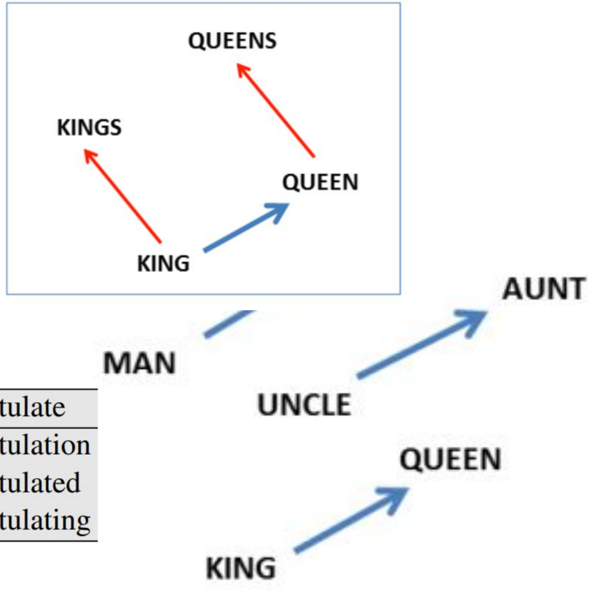
Yoshua Bengio, Réjean Ducharme, Pascal Vincent, Christian Jauvin A Neural Probabilistic Language Model, JMLR 3(Feb):1137-1155, 2003.

Let’s grumble

2013. Google’s researchers team publishes a paper describing a novel word vectors representations training algorithm, demonstrating that vectors

1) allow to estimate words similarity reasonably well
2) preserve some **relations** as vector subtraction

Thus, thanks to Google’s PR-machine all the coders (even without any linguistic background or interest) around the world now know what distributional semantics is :)



target:	Redmond	Havel	ninjutsu	graffiti	capitulate
	Redmond Wash.	Vaclav Havel	ninja	spray paint	capitulation
	Redmond Washington	president Vaclav Havel	martial arts	grafitti	capitulated
	Microsoft	Velvet Revolution	swordsmanship	taggers	capitulating

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space // In Proceedings of Workshop at ICLR, 2013
Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representations // In Proceedings of NAACL HLT, 2013

word2vec is a family of algorithms

SGNS: Skip-grams with Negative Sampling

predicting ‘window contexts’ given the word

CBOW: Continuous Bag-of-Words

predicting the word given the ‘window context’ (won’t discuss)

inb4 -- T. Mikolov:

Skip-gram: works well with small amount of the training data, represents well even rare words or phrases.

CBOW: several times faster to train than the skip-gram, slightly better accuracy for the frequent words

https://www.quora.com/What-are-the-continuous-bag-of-words-and-skip-gram-architectures

skip-grams

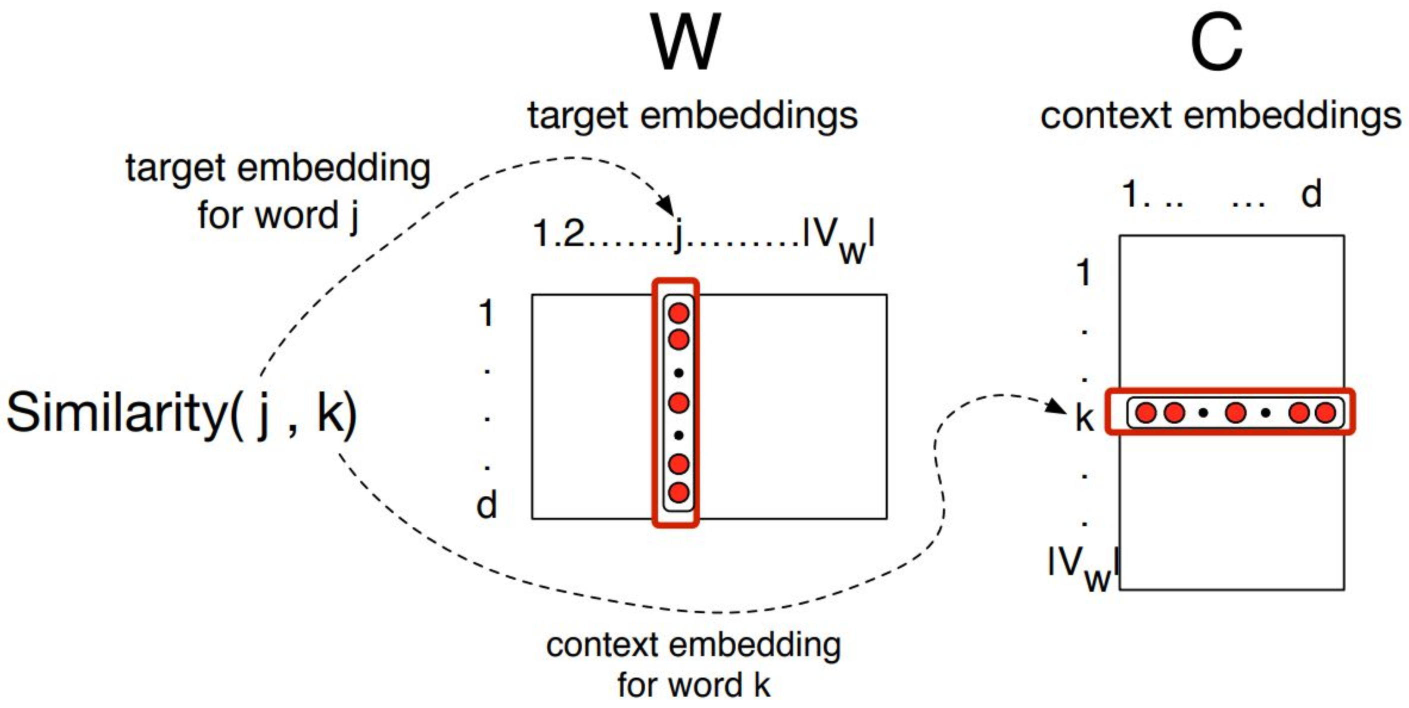
Scanning the text with **2L**-word window and learning to predict context words for the current word; that is, given the word $\mathbf{w_t}$ we estimate the probabilities of its occurrence close to the words $\mathbf{w_{t-L}w_{t-L+1}...w_{t-1}w_{t+1}...w_{t+L}}$.

Prediction - then correction **based on divergence from true values** -
- prediction - correction - …

Core steps:

1) Each word and each context are paired with a dense vector (initially a random one)
2) Word and context similarity score -- their vectors' scalar product
3) We train vectors values so that $\mathbf{p(v_{context}|v_{word})}$ (computed based on scalar product (2)) for correct contexts were larger

skip-grams


W
target embeddings
C
context embeddings
target embedding
for word j
1.2…….j………|Vw|
1
.
.
.
d
1. ..   … d
1
.
.
k
.
.
|Vw|
Similarity(j , k)
context embedding
for word k

skip-grams

We’ve measured similarity with cosine distance before and we know it can be treated as ‘normalized scalar product’; we want a similar thing here:

$$\text{Similarity}(j,k) \propto c_k \cdot v_j$$

...but we need probabilities. Then **softmax** is for us

$$p(w_k|w_j) = \frac{exp(c_k \cdot v_j)}{\sum_{i \in |V|} exp(c_i \cdot v_j)}$$

BTW, a problem: a sum of |V| scalar products in the denominator (time-consuming!)
Can be solved with **negative sampling** or **hierarchical softmax**

skip-grams with negative sampling

Computing one probability with **|V|m** multiplication and **|V|(m - 1)** addition ops, and computing **|V|+1** exponent function values is way too expensive

Things can be simplified:

1. maximization of scalar products sigmoids with the **true contexts**,
2. minimization if scalar products sigmoids with **random contexts** (this is what is called here **negative samples**)

$$\sigma(x) = \frac{1}{1+e^{x}}$$

skip-grams with negative sampling

$\sigma(x) = \frac{1}{1+e^x}$

Let's say we have window of size 2 -- 'positive' contexts

```
lemon,  a [tablespoon of apricot preserves or] jam
              c1        c2    w       c3        c4
```

we want to increase this

$$\sigma(c1 \cdot w) + \sigma(c2 \cdot w) + \sigma(c3 \cdot w) + \sigma(c4 \cdot w)$$

k = 2 means the fraction of 'negative' contexts is 1:2

```
[cement metaphysical dear coaxial
 n1     n2           n3   n4
apricot attendant whence forever puddle]
        n5        n6     n7      n8
```

we want to decrease this

$$\sigma(n1 \cdot w) + \sigma(n2 \cdot w) + ... + \sigma(n8 \cdot w)$$

skip-grams with negative sampling

Let’s write down the error for every word-context pair

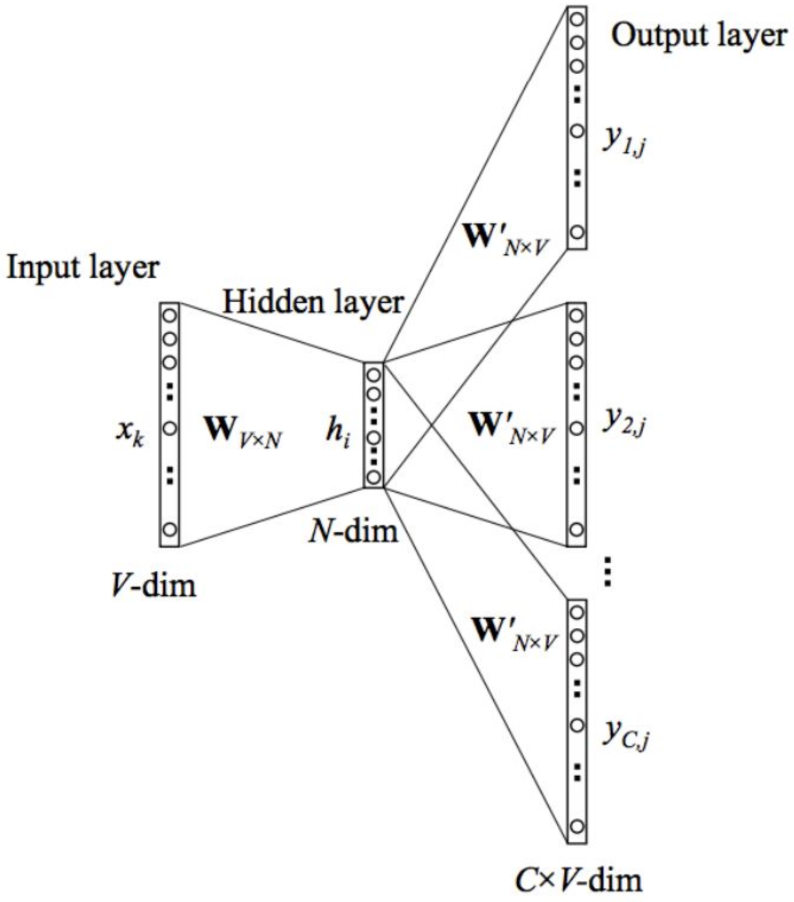
$$\log \sigma(c \cdot w) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim p(w)} [\log \sigma(-w_i \cdot w)]$$

This is not a SoftMax, but it works

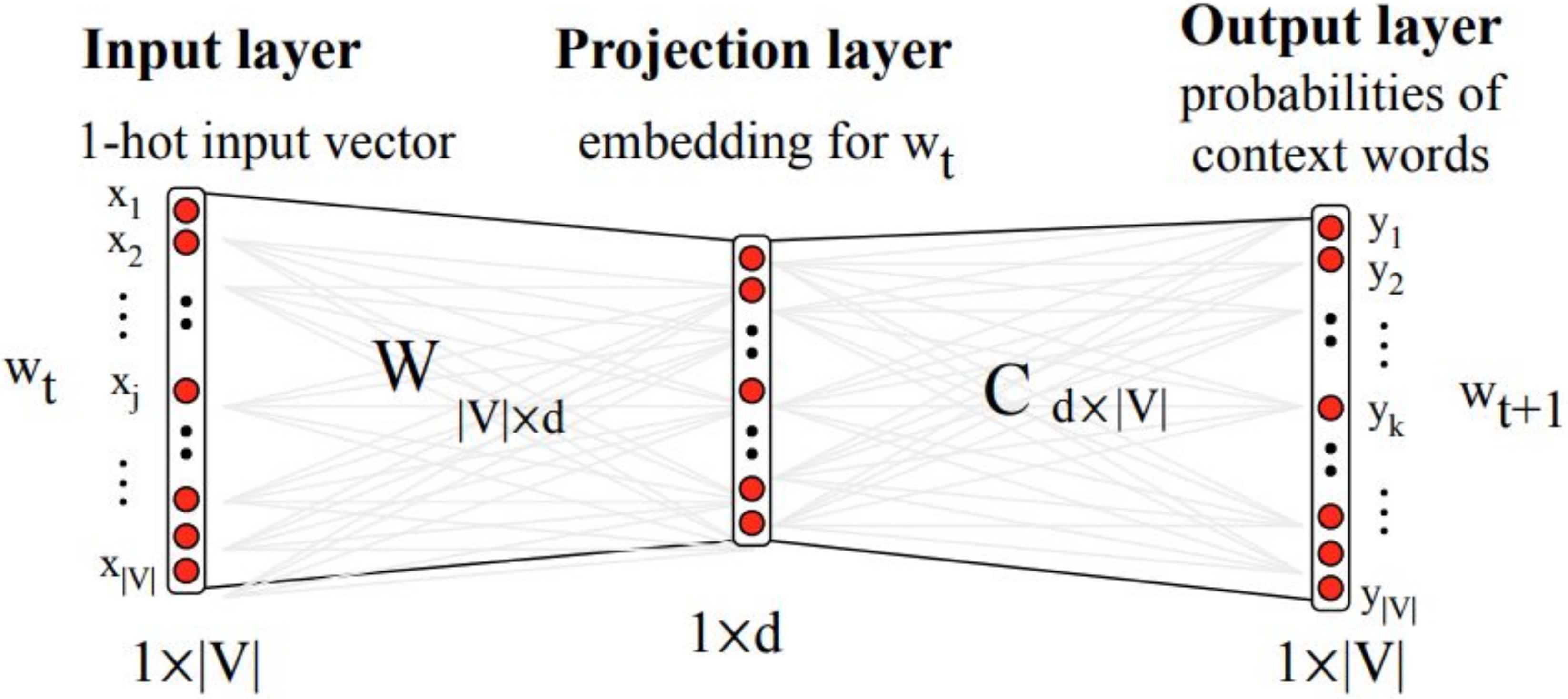
Neural network-like view

Training with backpropagation (BackProp)

(see tutorial или one more)



Neural network-like view



Connection with matrix factorization

It is proved that when skip-gram reaches optimumn the following holds:

$$WC = X^{\mathrm{PMI}} - \log k$$

Which implies that **word2vec** is an implicit matrix factorization of the sparse PMI word-context matrix!

But still it works better. Why?

- Introduces many **engineering tweaks** and **hyperpararameter settings**
  - May seem minor, but **make a big difference** in practice
  - Their impact is often more significant than the embedding algorithm's

Levy, O. and Goldberg, Y. Neural word embedding as implicit matrix factorization. In NIPS 14, pp. 2177– 2185.

Tools

Many open implementations, mainstream ones are

- **gensim**
- **word2vec** (от Google)
- GloVE (Stanford)
- fastText (FacebookAIResearch)
- implementations in popular NN frameworks

Pretrained vectors for different languages, e.g.

- RusVectores
- Not sure if this list is complete and/or good
  (however, you can always google vectors for your language of interest)

Datasets

For English

WordSim-353 - 353 noun pairs with ‘similarity scores’ estimates from 0 to 10

SimLex-999 - similar task with different parts-of-speech + synonymy is important

TOEFL dataset - 80 quizzes: a word + four more, the task is to choose a synonym

Also there are datasets where contexts are also available

For Russian

Translations of standard datasets + thesauri data

https://github.com/nlpub/russe-evaluation

Also see

Other popular vector representations

Glove: J. Pennington, R. Socher, C. Manning. Global Vectors for Word Representation EMNLP2014
fastText: P. Bojanowski, E.Grave, A. Joulin,T. Mikolov. Enriching word vectors with subword information, 2016.

Text representations

doc2vec: Le Q., Mikolov T. Distributed representations of sentences and documents // ICML-14

Handling polysemy:

AdaGram: S. Bartunov, D. Kondrashkin, A. Osokin, D. Vetrov. Breaking Sticks and Ambiguities with Adaptive Skip-gram. International Conference on Artificial Intelligence and Statistics (AISTATS) 2016.

And many more...

Used/recommended materials

1. Martin/Jurafsky, Ch. 15
2. Yoav Goldberg: word embeddings what, how and whither
3. Papers on slides
4. Valentin Malykh from ODS/iPavlov on w2v
5. A very cool explanation of what word2vec is
6. Wikipedia

Vector semantics

Anton Alekseev,
Steklov Mathematical Institute in St Petersburg

NRU ITMO, St Petersburg, 2019
anton.m.alexeyev+itmo@gmail.com

Many thanks to Denis Kirjanov for words of advice

On entropy of sequences
and its connection with perplexity

please see Martin/Jarfsky ed.3 **4.7**
https://web.stanford.edu/~jurafsky/slp3/4.pdf

additional slides on that are in Russian

Энтропия последовательности

- Часто нам важен текст как последовательность
- Нет проблем: для всякого языка L, задающего последовательности длины n

$$H(w_1, ..., w_n) = - \sum_{(w_1, ..., w_n) \in L} p(w_1, ..., w_n) log_2 p(w_1, ..., w_n)$$

- Энтропия языка с последовательностями бесконечной длины

$$H(L) = \lim_{n \to \infty} \frac{1}{n} H(w_1, ...w_n) =$$

$$= - \lim_{n \to \infty} \frac{1}{n} \sum_{(w_1, ..., w_n) \in L} p(w_1, ..., w_n) log_2 p(w_1, ..., w_n)$$

Энтропия последовательности

- Часто нам важен текст как последовательность
- Нет проблем: для всякого языка L, задающего последовательности длины n

$$H(w_1, ..., w_n) = - \sum_{(w_1, ..., w_n) \in L} p(w_1, ..., w_n) log_2 p(w_1, ..., w_n)$$

- Энтропия языка с последовательностями бесконечной длины

$$H(L) = \lim_{n \to \infty} \frac{1}{n} H(w_1, ... w_n) =$$

$$= - \lim_{n \to \infty} \frac{1}{n} \sum_{(w_1, ..., w_n) \in L} p(w_1, ..., w_n) log_2 p(w_1, ..., w_n)$$

УЖАС, как это считать?!

Стационарность стохастического процесса

Стохастический процесс называется **стационарным**, если вероятности последовательностей инвариантны относительно сдвигов позиций во времени

Википедия

- Случайный процесс называется *стационарным*, если все многомерные законы распределения зависят только от взаимного расположения моментов времени $t_1, t_2, \ldots, t_n$, но не от самих значений этих величин. Другими словами, случайный процесс называется стационарным, если его вероятностные закономерности неизменны во времени. В противном случае, он называется *нестационарным*.

Для естественного языка это, очевидно, не так, но иногда в рамках моделей мы можем себе позволить такое приближение

Эргодический стационарный стохастический процесс

В. Д. Колесник, Г. Ш. Полтырев
“Курс теории информации”

Пусть U_X — стационарный источник, выбирающий сообщения из множества X, и $\ldots x^{(-1)}, x^{(0)}, x^{(1)}, x^{(2)}, \ldots$ — последовательность сообщений на его выходе. Пусть $\varphi(x_1, \ldots, x_k)$ — произвольная функция, определенная на множестве X^k и отображающая отрезки сообщений длины k в числовую ось. Пусть

$$z^{(i)} \triangleq \varphi(x^{(i+1)}, \ldots, x^{(i+k)}), \quad i = 1, 2, \ldots, \tag{1.9.8}$$

— последовательность случайных величин, имеющих в силу стационарности одинаковые распределения вероятностей. Обозначим через m_z математическое ожидание случайных величин $z^{(i)}$.

Определение 1.9.1. Дискретный стационарный источник называется *эргодическим*, если для любого k, любой действительной функции $\varphi(x_1, \ldots, x_k)$, $\mathrm{M}\varphi(\cdot) < \infty$, определенной на X^k, любых положительных ε и δ найдется такое N, что для всех $n > N$

$$\Pr\left(\left|\frac{1}{n}\sum_{i=1}^{n} z^{(i)} - m_z\right| \geq \varepsilon\right) < \delta. \tag{1.9.9}$$

Википедия

- Если при определении моментных функций стационарного случайного процесса операцию усреднения по статистическому ансамблю можно заменить усреднением по времени, то такой стационарный случайный процесс называется **эргодическим**.

Ответы Mail.RU

Попробую по-простому:
Помнишь что у случ процесс иногда записывают столбиком его реализации? Дак вот
можешь в любой момент времени провести сечение через неск-ко реализаций, найти
среднее значение, и оно окажется таким же, как если бы ты усреднял только одну
реализацию)))

Энтропия последовательности

- **Теорема Шэннона-МакМиллана-Breймана** спешит на помощь: при стационарности и эргодичности последовательности верно, что

$$H(L) = -\lim_{n\to\infty} \frac{1}{n} log_2 p(w_1, ...w_n)$$

...то есть мы можем *просто* взять достаточно длинную последовательность для хорошей оценки
- То же верно при таких же допущениях и для перекрёстной энтропи

$$H(p, q) = -\lim_{n\to\infty} \frac{1}{n} log_2 q(w_1, ...w_n)$$

Зачем всё это: энтропия и перплексия

- Вспомним

$$p(x_1, ..., x_n) = \prod_{i=1}^{n} p(x_i|x_1, ..., x_{i-1})$$

- Выпишем формулу перплексии

$$PP(W) = \sqrt[n]{\frac{1}{p(x_1, ..., x_n)}} = 2^{-\frac{1}{n} log_2 p(x_1,...,x_n)} =$$

$$= 2^{-\frac{1}{n}\sum_{i=1}^{n} \log P(x_i|x_1...x_{i-1})} \rightarrow 2^{H(W)}$$

- Перплексия — экспонента кросс-энтропии языка, которую мы оцениваем на достаточно длинном тексте