ITMO UNIVERSITY

Saint Petersburg, Russia

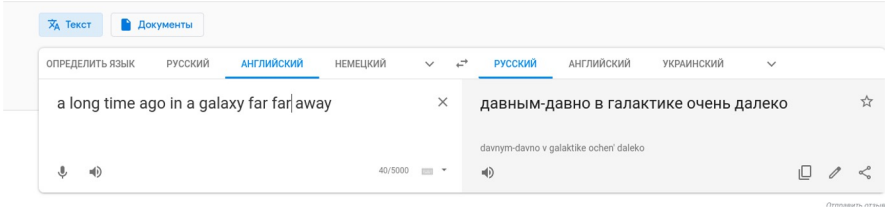
Natural Language Processing (NLP)

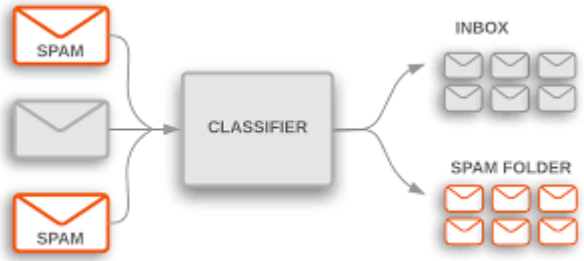
Severiukhina Oksana
PhD, ex. Senior Researcher at NCCR

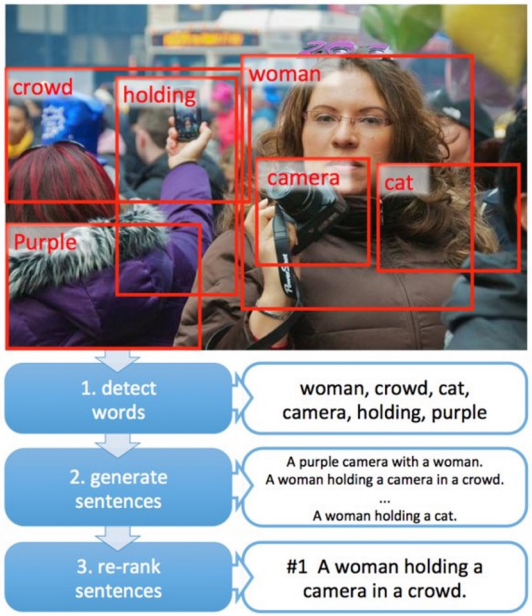
2023, February 17

Natural Language processing tasks

- machine translation
- annotation
- text classification
- image captioning
- text generation
- chat bots
- dialog systems
- and so on







Natural Language Processing

ITMO UNIVERSITY

NLP is a subfield of **linguistics**, **computer science**, and **artificial intelligence** concerned with the interactions between computers and human language, in particular how to program computers to process and analyse large amounts of **natural language** data.

Wikipedia

Agenda

1. Text data and its preprocessing
2. Classification of approaches
3. Frequency models
4. Vector similarity metrics
5. Word embeddings (Word2Vec, Glove, FastText)
6. State-of-the-art approaches

ITMO UNIVERSITY

Intro

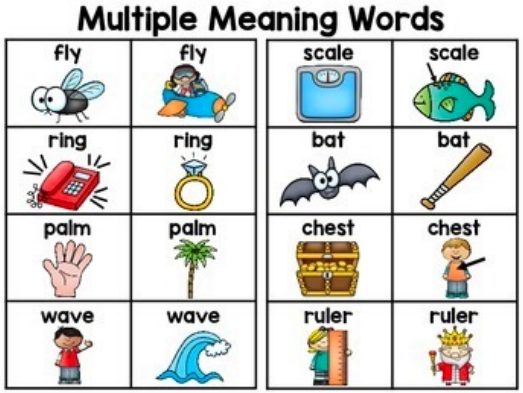
Why is NLP difficult?

ITMO UNIVERSITY

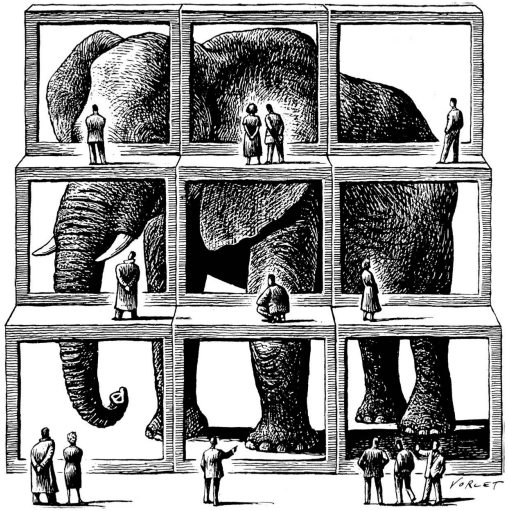
Symbolic representation (can be expressed in several ways or languages)

東 = est = ESTE = **east**
西 = ouest = OESTE = **west**
北 = nord = NORTE = **north**
南 = sud = SUR = **south**

Homonyms



Context matters



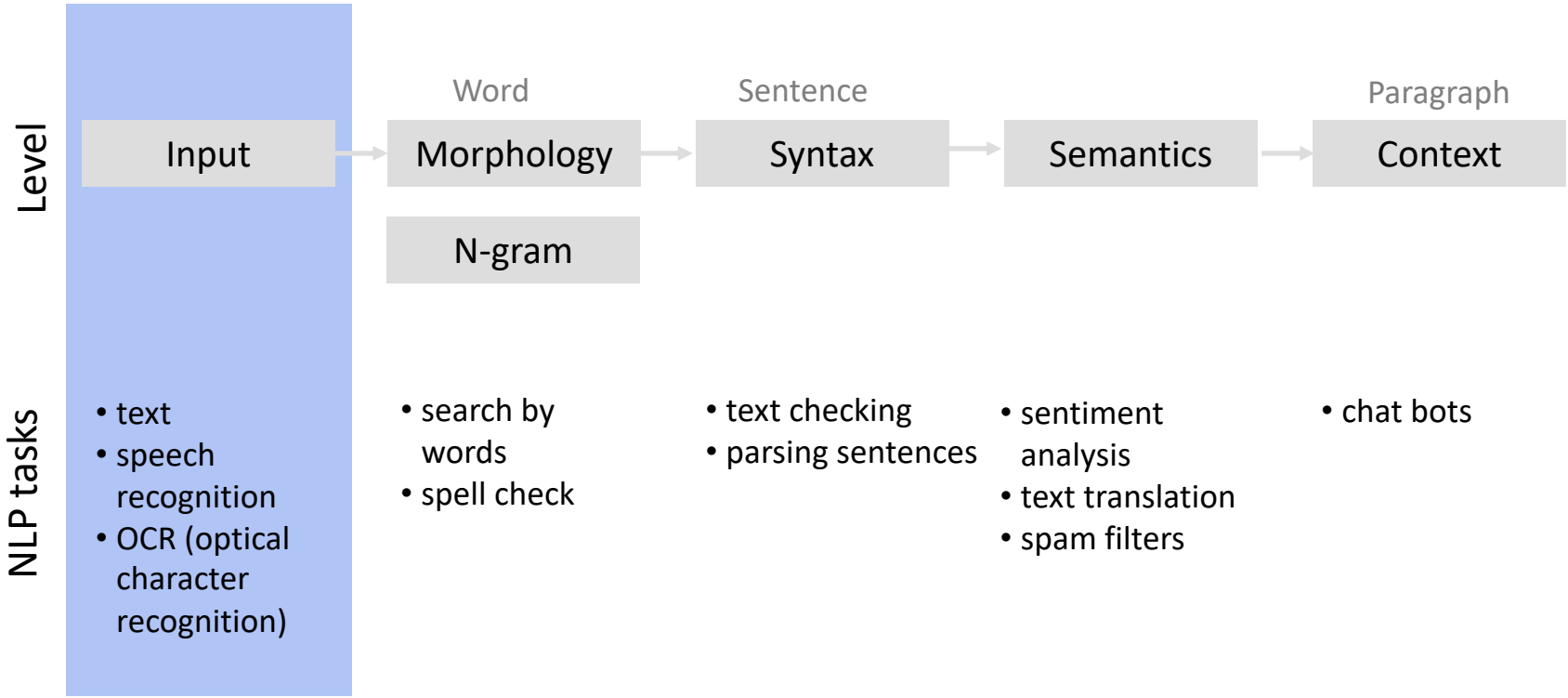
What is text?

ITMO UNIVERSITY

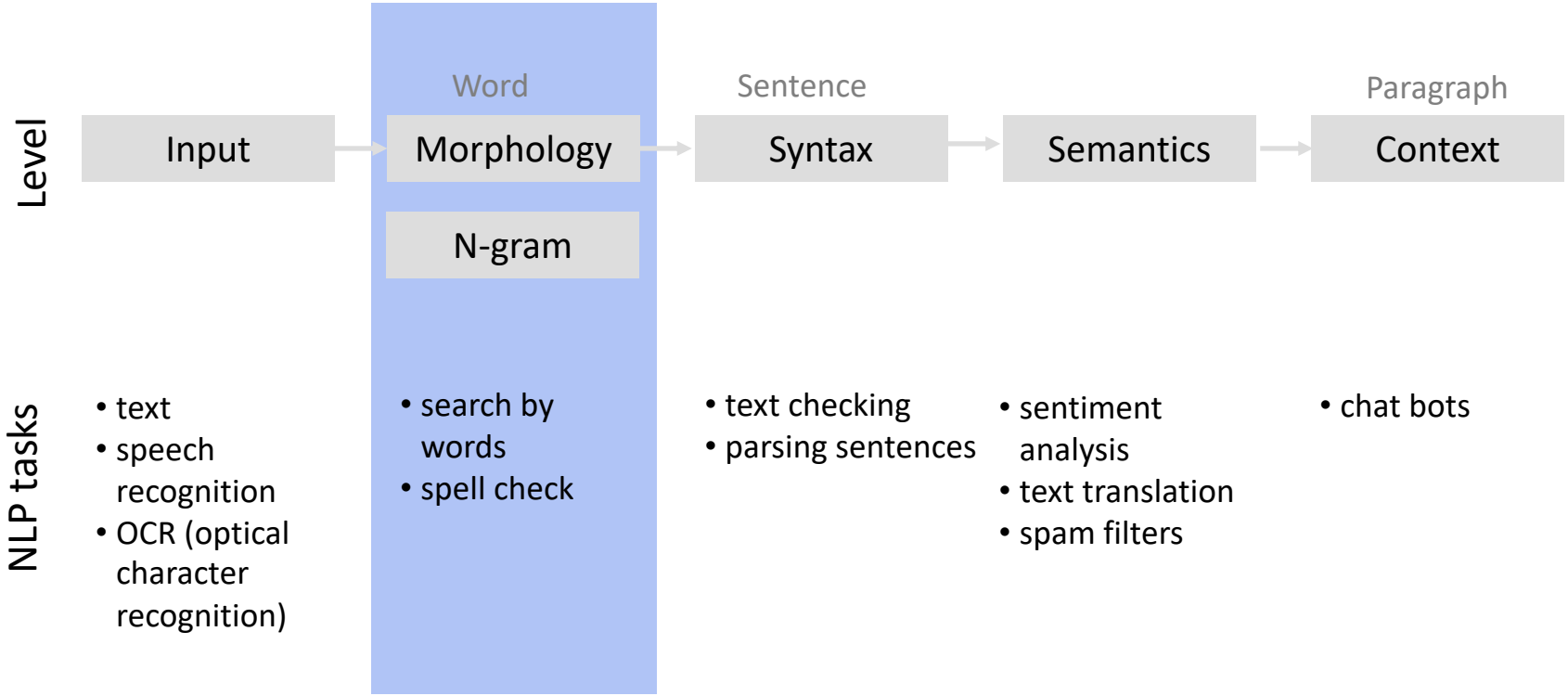
The text can be thought of as **a sequence of** :

- Characters
- Words
- Phrases and named entities
- Sentences
- Paragraphs
- …

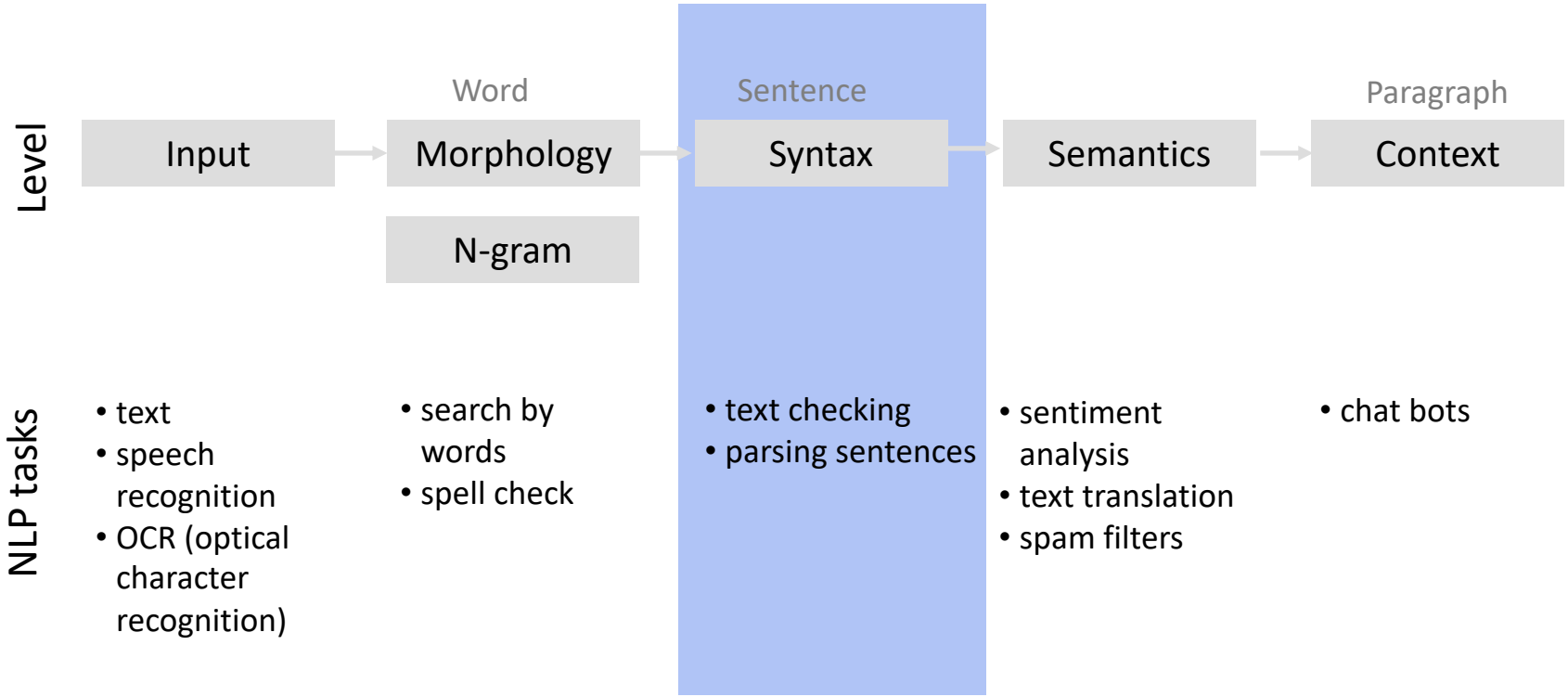
NLP tasks


Level
Input
Word
Morphology
N-gram
Sentence
Syntax
Semantics
Paragraph
Context
NLP tasks
• text
• speech recognition
• OCR (optical character recognition)
• search by words
• spell check
• text checking
• parsing sentences
• sentiment analysis
• text translation
• spam filters
• chat bots

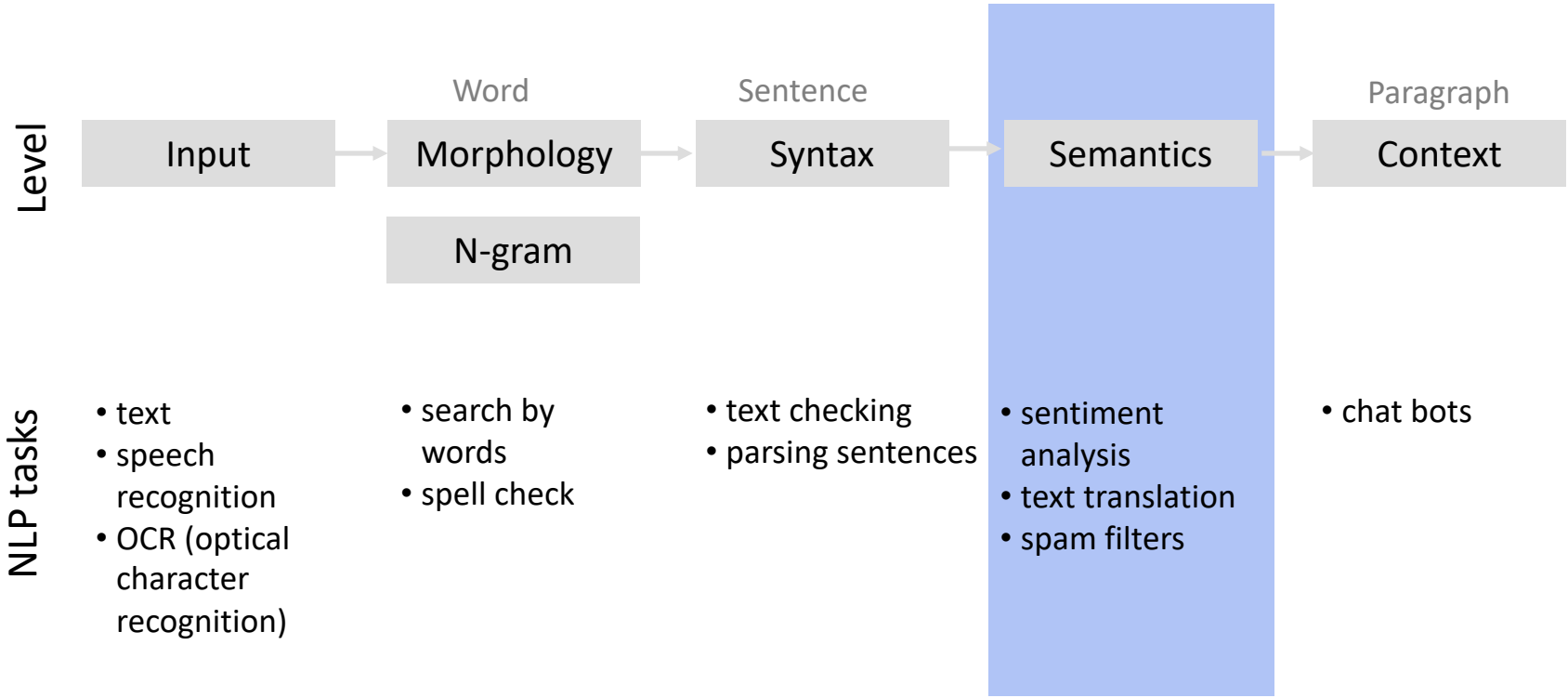
NLP tasks


Level
Input
Word
Morphology
N-gram
Sentence
Syntax
Semantics
Paragraph
Context
NLP tasks
• text
• speech recognition
• OCR (optical character recognition)
• search by words
• spell check
• text checking
• parsing sentences
• sentiment analysis
• text translation
• spam filters
• chat bots

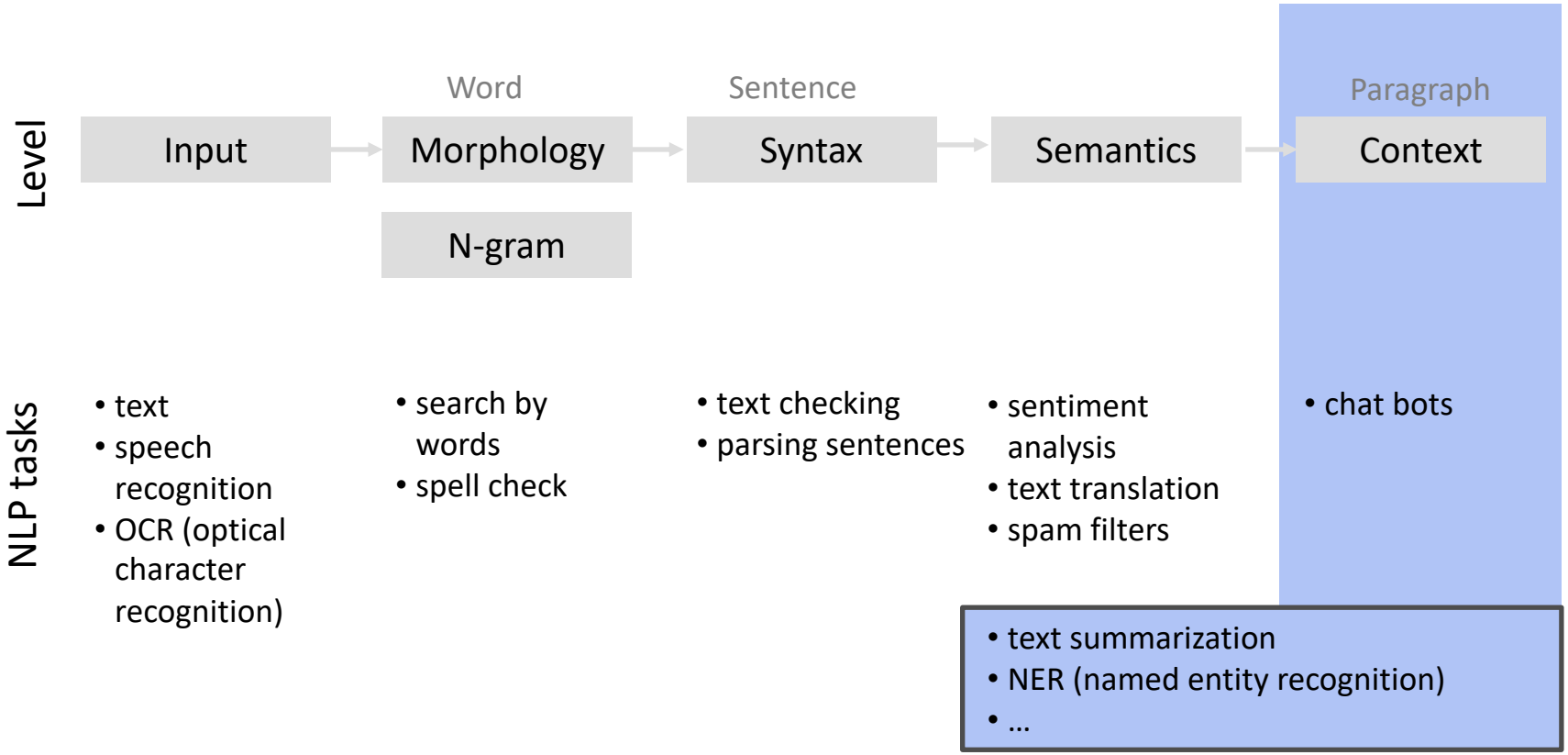
NLP tasks


Level
Input
Word
Morphology
N-gram
Sentence
Syntax
Semantics
Paragraph
Context
NLP tasks
• text
• speech recognition
• OCR (optical character recognition)
• search by words
• spell check
• text checking
• parsing sentences
• sentiment analysis
• text translation
• spam filters
• chat bots

NLP tasks


Level
Word
Sentence
Paragraph
Input
Morphology
Syntax
Semantics
Context
N-gram
NLP tasks
• text
• speech recognition
• OCR (optical character recognition)
• search by words
• spell check
• text checking
• parsing sentences
• sentiment analysis
• text translation
• spam filters
• chat bots

NLP tasks


Level
Word
Sentence
Paragraph
Input
Morphology
Syntax
Semantics
Context
N-gram
NLP tasks
• text
• speech recognition
• OCR (optical character recognition)
• search by words
• spell check
• text checking
• parsing sentences
• sentiment analysis
• text translation
• spam filters
• chat bots
• text summarization
• NER (named entity recognition)
• …

Solving NLP task step by step:

ITMO UNIVERSITY

Step 1:

Gather the data

Step 2:

Clean and preprocess the data

Step 3:

Fit the model (classification, clustering, and so on)
and inspect the results by some metric

Step 4:

Use more sophisticated representation models
(word2vec, fastTest etc.)

Basic definitions

ITMO UNIVERSITY

Corpus is a set of text data. Consists of unformatted text and metadata corresponding to the text.

Unformatted text consists of characters that can be grouped into contiguous data units called **tokens.** Tokens can match a sequence of words and a number separated by spaces.

Metadata is supporting information related to a given text. May include identifiers, labels, time/date.

Vocabulary is the set of all tokens in the corpus.

“Your model will only ever be as good as your data.”

ITMO UNIVERSITY

Data preprocessing

Data preprocessing

ITMO UNIVERSITY

A clean dataset will allow a model to learn meaningful features and not overfit on irrelevant noise.

Checklist:

1. Remove all **irrelevant characters** such as any non alphanumeric characters;
2. **Tokenize** your text by separating it into individual words; → word == "token"
3. Remove words that are not relevant, such as "@" twitter mentions or URLs (~remove **stopwords**);
4. Convert all characters to **lowercase**, in order to treat words such as "hello", "Hello", and "HELLO" the same
5. Consider combining misspelled or alternately spelled words to a **single representation** (e.g. "cool"/"kewl"/"cooool")
6. Consider normalization: **lemmatization** (reduce words such as "am", "are", and "is" to a common form such as "be") or **stemming**

How to solve 90 of nlp problems a step-by-step guide

Tokenization

ITMO UNIVERSITY

The process of dividing the input sequence into parts (tokens).

Example:

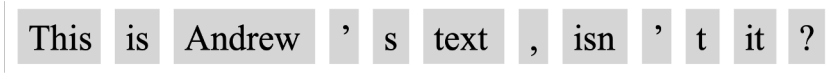
This is Andrew’s text, isn’t it?

1. Split by space (nltk.tokenize.WhitespaceTokenizer)

- “it” and “it?” tokens are different, but have the same meaning

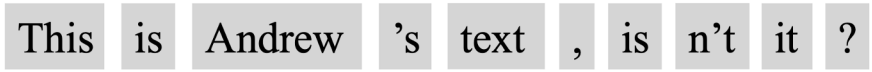
2. Separation by punctuation (nltk.tokenize.WordPunctTokenizer)

- “s”, “isn”, “t” tokens don’t make sense



3. Separation with rules (nltk.tokenize.TreebankWordTokenizer)

- “’s” и “n’t” tokens make sense when processing



Normalization

ITMO UNIVERSITY

We want the same token for different forms of the word:

- wolves and wolf
- talks and talk

Stemming

A process of removing and replacing suffixes to get to the root form of the word, which is called the stem.

Usually refers to heuristics that chop off suffixes.

Lemmatization

Usually refers to doing things properly with the use of a vocabulary and morphological analysis

Returns the base or dictionary form of a word, which is known as the lemma.

Stemming example

ITMO UNIVERSITY

Porter Stemmer

Disadvantage:

- produces non-words
- fails on irregular forms

nltk.stem.PorterStemmer

- feet -> feet
- wolves-> wolv
- cats -> cat
- talked -> talk

Examples of rules:

Rule	Example
SSES → SS	caresses → caress
IES → I	ponies → poni
SS → SS	caress → caress
S →	cats → cat

Lemmatization example

ITMO UNIVERSITY

WordNet Lemmatiser

Uses WordNet Databese to lookup lemmas

Disadvantage:

- not all forms are reduced

nltk.stem.WordNetLemmatizer

- feet -> foot
- wolves-> wolf
- cats -> cat
- talked -> talked

WordNet is a lexical database of semantic relations between words i n more than 200 languages.

Try stemming and lemmatization, then choose the best for the current task

A little more about normalization and preprocessing

ITMO UNIVERSITY

Normalizing capital letters

Us, us -> us (if both are pronoun)

us, US (could be pronoun and country)

We can use heuristics:

- lowercasing the beginning of the sentence, lowercasing words in titles
- leave mid-sentence words as they are

Or we can use machine learning to retrieve true **Acronyms**

eta, e.t.a., E.T.A. –> E.T.A.

In social network:

- remove all hashtags/# sign
- remove mentions of @
- Replace any emojis with the text they represent:

SMILEYS = {":-(":"sad", ":-)":"smiley", ...}

ITMO UNIVERSITY

Approaches

Classification of approaches

ITMO UNIVERSITY

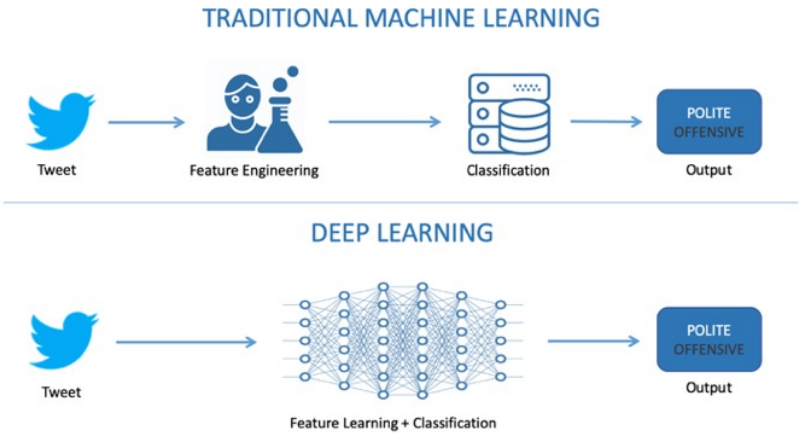
Type	Rule-based methods	Statistical NLP (frequency)	Vector methods	Deep Learning
Examples	• Regular expressions • Context-free grammars	• BoW • TF-IDF	• Word2vec • fastText	• Recurrent Neural Networks • BERT
Some properties	• a lot of time to set the rules • Explicit logic • Poor scaling • High precision and low recall	• features generation • No semantic capture • Does not generalize	• Capture semantic • No capture of document structure	• Require huge training set

Traditional algorithms vs DL

ITMO UNIVERSITY

Why study traditional NLP algorithms when there are DL?

1. Perform well in many tasks: most tasks do not require DL and large datasets.
2. Do not follow the hype blindly.
3. Help improve DL methods.



N-grams

N-gram is simply a collection of **n consecutive words** in a text.

Example:
“Sets of consecutive tokens”

n = 1 - **unigrams**: sets, of, consecutive, tokens
n = 2 - **bigrams**: Sets of, of consecutive, consecutive tokens
n = 3 - **trigrams**: “sets of consecutive” and “of consecutive tokens”

Syntactic n-grams are *n*-grams defined by paths in syntactic dependency or constituent trees rather than the linear structure of the text

For example, the sentence "economic news has little effect on financial markets" can be transformed to syntactic *n*-grams following the tree structure of its dependency relations:

- news-economic,
- effect-little,
- effect-on-markets-financial.

Sidorov G at al "Syntactic Dependency-Based N-grams as Classification Features"

ITMO UNIVERSITY

Statistical NLP

Bag-of-Words approach (BoW)

ITMO UNIVERSITY

It creates a vocabulary of all the unique words occurring in all the documents in the training set.

For example, if you have 3 documents:

D1 - “I am feeling very happy today”;
D2 - “I am not well today”;
D3 - “I wish I could go to play”;

It creates a vocabulary using unique words from all the documents.
Unique list of words:

I am feeling very happy today not well wish could go to play

For each word the frequency of the word in the corresponding document is calculated:

	I	am	feeling	very	happy	today	not	well	wish	could	go	to	play
D1	1	1	1	1	1	1	0	0	0	0	0	0	0
D2	1	1	0	0	0	1	1	1	0	0	0	0	0
D3	2	0	0	0	0	0	0	0	1	1	1	1	1

Each index in the vector corresponds to one word.

TF – IDF approach

ITMO UNIVERSITY

Term Frequency – Inverse Document Frequency.

The approach is intended to reflect how important a word is to a document in a collection or corpus;

$$tf(t,d) = \frac{n_t}{\sum_k n_k}$$

$$idf(t,D) = \log \frac{|D|}{|\{d_i \in D | t \in d_i\}|}$$

$$tf\_idf(t,d,D) = tf(t,d) \times idf(t,D)$$

n_t – the frequency of the token t in the document d

$|D|$ – the number of the documents in the collection

$|d_i \in \mathrm{D} | t \in d_i|$- the number of the documents in collection where t occurs

TF – IDF approach: example

Documents	Text	Total number of words in a document
A	Jupiter is the largest planet	5
B	Mars is the fourth planet from the sun	8

ITMO UNIVERSITY

Words	TF (for A)	TF (for B)	IDF	TFIDF (A)	TFIDF (B)
Jupiter	1/5	0	ln(2/1) = 0.69	0.138	0
Is	1/5	1/8	ln(2/2) = 0	0	0
The	1/5	2/8	ln(2/2) = 0	0	0
largest	1/5	0	ln(2/1) = 0.69	0.138	0
Planet	1/5	1/8	ln(2/2) = 0	0.138	0
Mars	0	1/8	ln(2/1) = 0.69	0	0.086
Fourth	0	1/8	ln(2/1) = 0.69	0	0.086
From	0	1/8	ln(2/1) = 0.69	0	0.086
Sun	0	1/8	ln(2/1) = 0.69	0	0.086

TF – IDF approach

ITMO UNIVERSITY

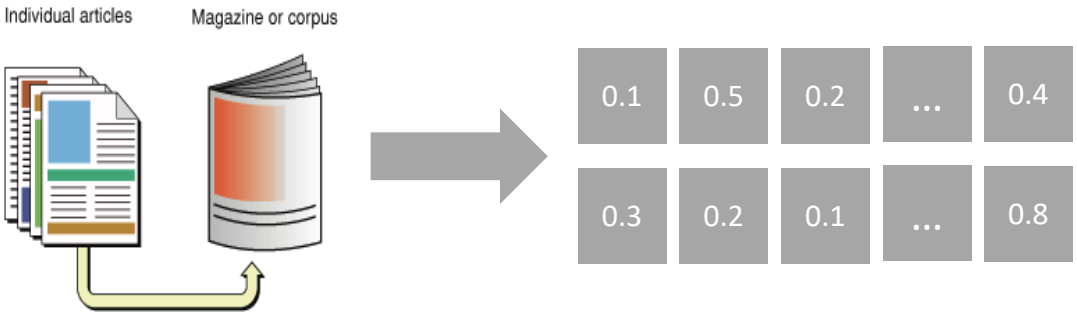
Large **TF-IDF** will get words with a **high frequency** within a specific document and with a **low frequency** of use in other documents.

The **TF-IDF** measure is often used to represent a collection of documents in the form of numerical vectors that reflect the importance of using each word from a certain set of words in each document.

$$tf(t,d) = \frac{n_t}{\sum_k n_k}$$

$$idf(t,D) = \log \frac{|D|}{|\{d_i \in D | t \in d_i\}|}$$

$$tf\_idf(t,d,D) = tf(t,d) \times idf(t,D)$$



Comparing by different distance / similarity measures

Vector similarity metrics

ITMO UNIVERSITY

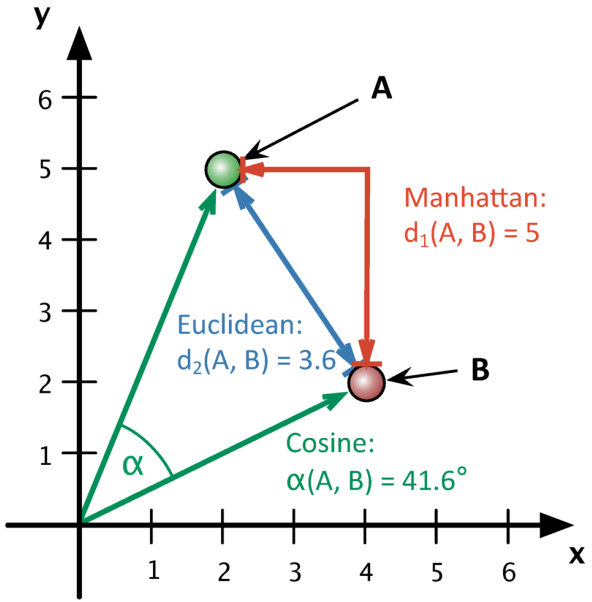
For two vectors: $a = (a_1, \dots, a_n), b = (b_1, \dots, b_n),$

1. Manhattan distance:

$$d(a,b) = \sum_{i=1}^{n} |a_i - b_i|$$

2. Euclidean distance :

$$d(a,b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$



d – **a distance measure (a metric)** on a set X

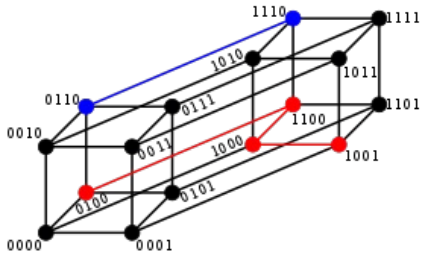
Properties

1. d is positive definite:
   $d(x,y) \geq 0,$
   $d(x,y) = 0 \Leftrightarrow x = y$
2. d is symmetric:
   $d(x,y) = d(y,x)$
3. d is subadditive (triangle
   $(x,z) \leq d(x,y) + d(y,z)$

Vector similarity metrics

For two vectors: $a = (a_1, \dots, a_n), b = (b_1, \dots, b_n)$,

3. Hamming distance is the number of positions at which the corresponding symbols are different.



Two example distances:
0100→1001 has distance 3;
0110→1110 has distance 1

d – **a distance measure** (**a metric**) on a set X

Properties

1. d is positive definite:
   $d(x, y) \geq 0,$
   $d(x, y) = 0 \Leftrightarrow x = y$
2. d is symmetric:
   $d(x, y) = d(y, x)$
3. d is subadditive (triangle
   $(x, z) \leq d(x, y) + d(y, z)$

4. Cosine similarity:

$$similarity(a, b) = \cos(\theta) = \frac{a \cdot b}{\|a\|\|b\|} = \frac{\sum_i^N a_i b_i}{\sqrt{\sum_i^N (a_i)^2}\sqrt{\sum_i^N (b_i)^2}}$$

For frequency methods (BoW, tf-idf), the cosine similarity of the two documents ranges from 0 to 1, since the frequencies cannot be negative. The angle between two vectors cannot be greater than 90.

Vector similarity metrics

ITMO UNIVERSITY

The Three Documents and Similarity Metrics

Considering only the 3 words from the above documents: 'sachin', 'dhoni', 'cricket'

Doc Sachin: Wiki page on Sachin Tendulkar	Doc Dhoni: Wiki page on Dhoni	Doc Dhoni_Small: Subsection of wiki on Dhoni
Dhoni - 10	Dhoni - 400	Dhoni - 10
Cricket - 50	Cricket - 100	Cricket - 5
Sachin - 200	Sachin - 20	Sachin - 1

Document - Term Matrix (Word Counts)

Word Counts	*"Dhoni"*	*"Cricket"*	*"Sachin"*
Doc Sachin	10	50	200
Doc Dhoni	400	100	20
Doc Dhoni_Small	10	5	1

Let's suppose you have 3 documents based on a couple of star cricket players – Sachin Tendulkar and Dhoni.
Two of the documents (A) and (B) are from the wikipedia pages on the respective players and the third document (C) is a smaller snippet from Dhoni's wikipedia page.

Cosine similarity

Vector similarity metrics

ITMO UNIVERSITY

The Three Documents and Similarity Metrics

Document - Term Matrix (Word Counts)

Word Counts	*"Dhoni"*	*"Cricket"*	*"Sachin"*
Doc Sachin	10	50	200
Doc Dhoni	400	100	20
Doc Dhoni_Small	10	5	1

Similarity or Distance Metrics	*Total Common Words*	*Euclidean distance*	*Cosine Similarity*
Doc Sachin & Doc Dhoni	10 + 50 + 20 = 80	432.4	0.15
Doc Dhoni & Doc Dhoni_Small	10 + 5 + 1 =16	401.9	0.97
Doc Sachin & Doc Dhoni_Small	10 + 5 + 1 =16	204.0	0.23

Cosine similarity

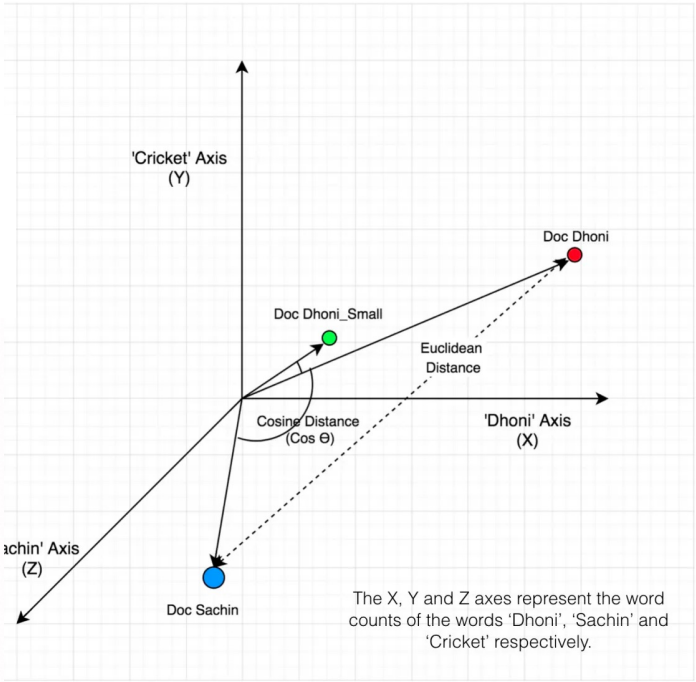
Vector similarity metrics

ITMO UNIVERSITY

The Three Documents and Similarity Metrics

Similarity or Distance Metrics	*Total Common Words*	*Euclidean distance*	*Cosine Similarity*
Doc Sachin & Doc Dhoni	10 + 50 + 20 = 80	432.4	0.15
Doc Dhoni & Doc Dhoni_Small	10 + 5 + 1 =16	401.9	0.97
Doc Sachin & Doc Dhoni_Small	10 + 5 + 1 =16	204.0	0.23

Projection of Documents in 3D Space



Cosine similarity

ITMO UNIVERSITY

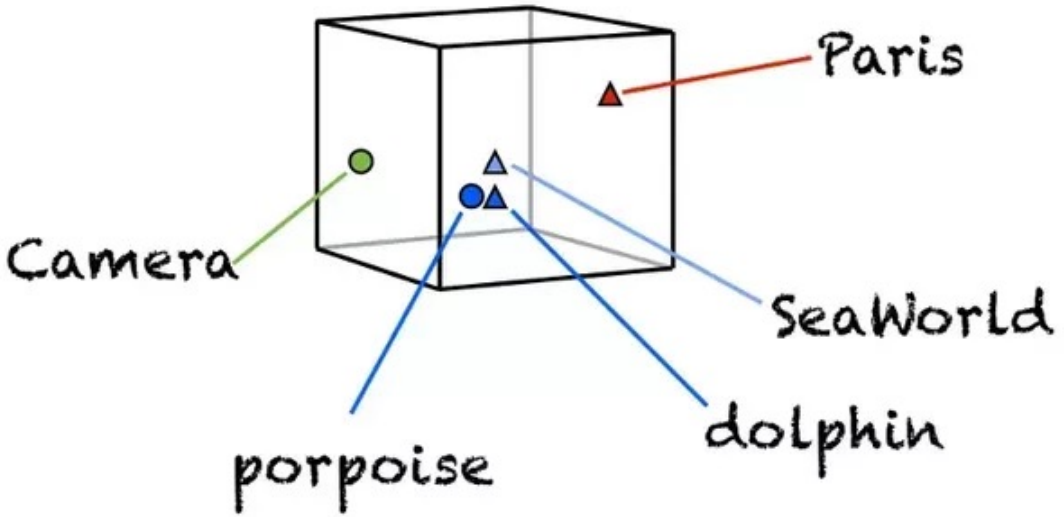
Vector methods

Word embedding

ITMO UNIVERSITY

Words with similar meanings will occur in similar contexts

- Representation of a word as vectors
- A vector defines a specific point in space



Context (1/3)

ITMO UNIVERSITY

What does the word **tezguino** mean?

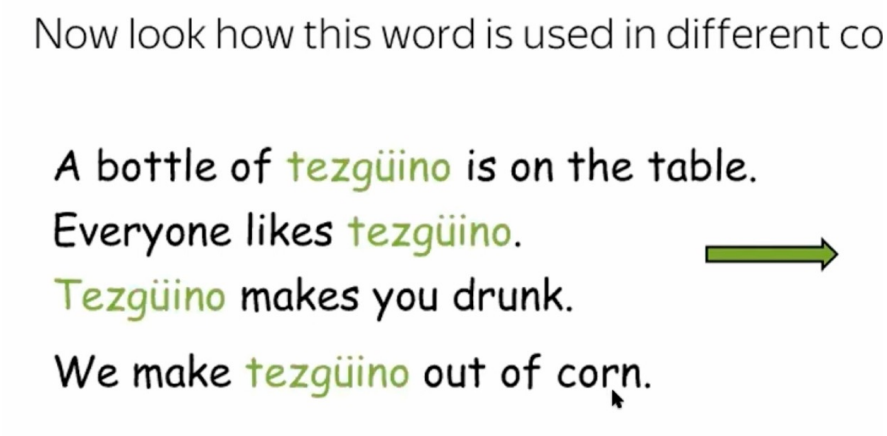


Illustration by Elena Voita

Context (2/3)

What does the word **tezguino** mean?

Now look how this word is used in different contexts:

A bottle of tezgüino is on the table.
Everyone likes tezgüino.
Tezgüino makes you drunk.
We make tezgüino out of corn.

Tezgüino is a kind of alcoholic beverage made from corn.

With context, you can understand the meaning!

Illustration by Elena Voita

Context (3/3)

(1) A bottle of _____ is on the table.

(2) Everyone likes _____ .

(3) _____ makes you drunk.

(4) We make _____ out of corn.

What other words fit into these contexts ?

	(1)	(2)	(3)	(4)	...
tezgüino	1	1	1	1	
loud	0	0	0	0	
motor oil	1	0	0	1	
tortillas	0	1	0	1	
wine	1	1	1	0	

← contexts

← rows show contextual properties: 1 if a word can appear in the context, 0 if not

Illustration by Elena Voita

Word embedding

Word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers.

one-hot representation

$$\begin{bmatrix} 0 \\ .. \\ 0 \\ 1 \\ 0 \\ ... \\ 0 \end{bmatrix}$$

distributed representation

$$\begin{bmatrix} 0.12 \\ 0.45 \\ -3.5 \\ -0.21 \\ 0.1 \\ -0.1 \\ 0.34 \end{bmatrix}$$

Word embedding vs One-hot encoding

ITMO UNIVERSITY

Both approaches are a way to turn a word into vector signs. But:

Dimension:

✗ Feature space in one-hot vectors has dimension equal to the number of words in collection dictionary (~ 10^3 -10^4).. This dimension is growing along with the growth of the vocabulary.

✓ Compressed vector representations are built in spaces of fixed dimension (~10^2).

Semantic:

✗ One-hot encoding does not take into account the semantic proximity of words, all vectors are equally far apart in an object space.

✓ Compressed vector representations for semantically close words are close as vectors (for example, by cosine similarity).

Word2Vec: idea

ITMO UNIVERSITY

The approach was proposed by T. Mikolov in 2013.
The original article:
Distributed Representations of Words and Phrases and their Compositionality

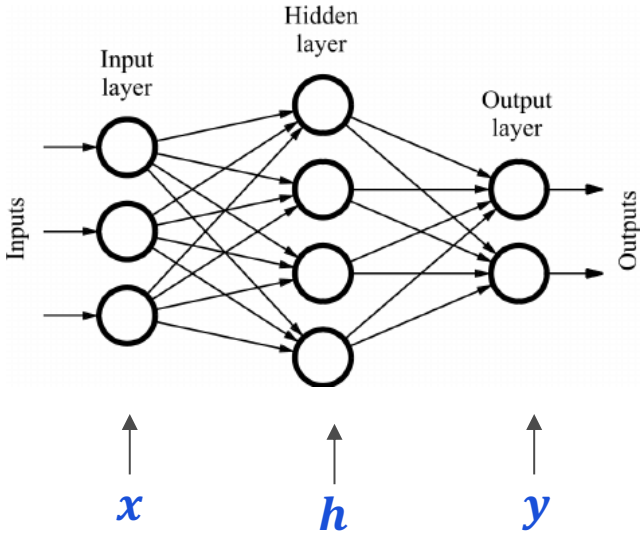
Tomas Mikolov

We're going to train a **simple neural network** with a single hidden layer, but then we're not actually going to use that neural network for the task we trained it on!

Instead, the goal is actually just to learn **the weights of the hidden layer.**

These weights are actually the **"word vectors"** that we're trying to learn.

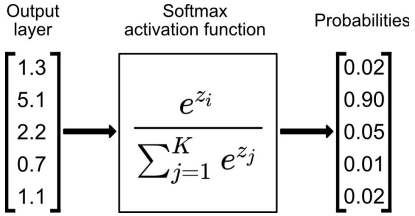
Neural network: base



h= $w_1 x$

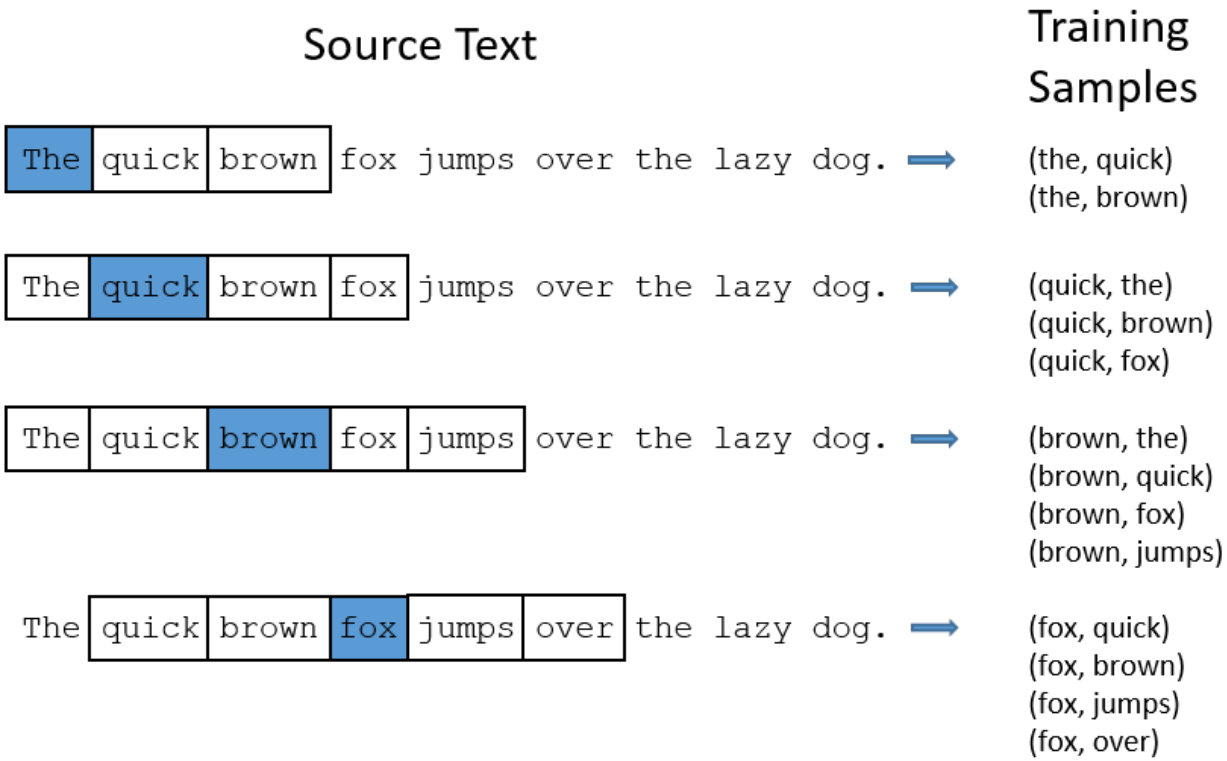
~~y= $w_2 h$~~

y= $softmax(w_2 h)$



Word and its context

ITMO UNIVERSITY



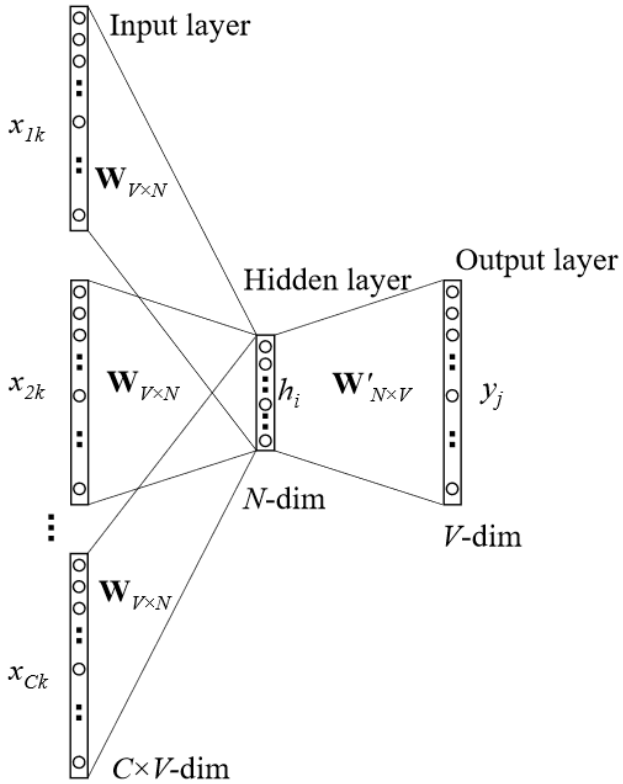
Word2vec-tutorial-the-skip-gram-model

Continuous bag-of-words model (CBOW)

CBOW predicts a word from the local context:

- inputs are one-hot representations of words of dimension V;
- the hidden layer is the matrix of word representations W;
- the output of the hidden layer is the average of the context word vectors;
- at the output, we get an estimate uj for each word and take softmax.

$$\hat{p}(i|c_1, \dots, c_n) = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})}$$



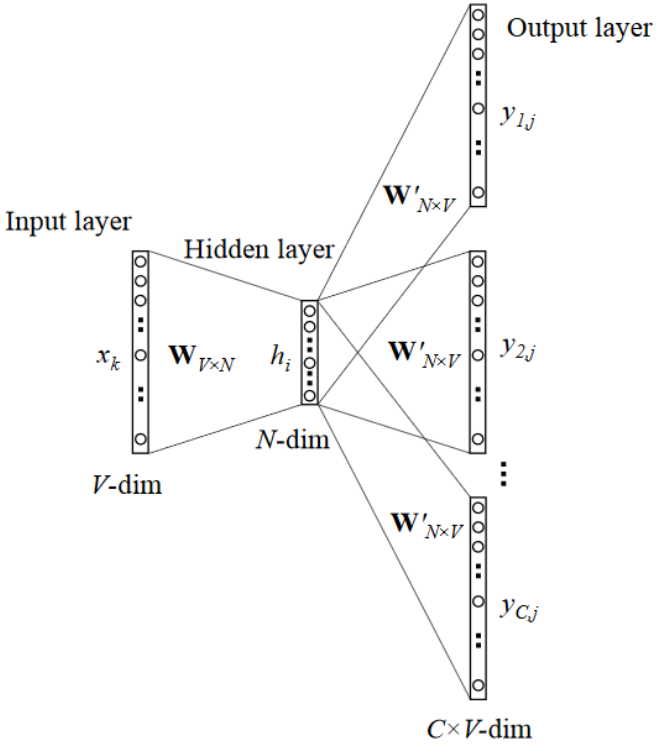
Image

Skip-gram

Skip-gram predicts context words from the current word:

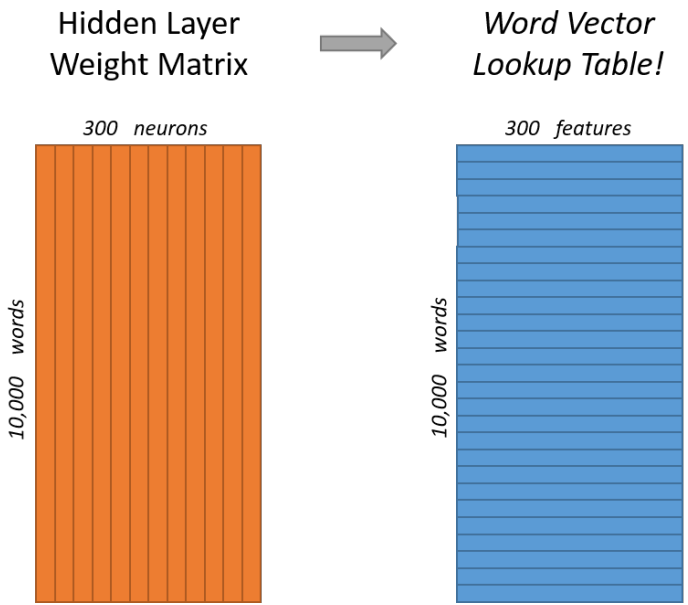
- predict each word of the context from the central one;
- now several multinomial distributions and softmax for each context word:

$$\hat{p}(c_k|i) = \frac{\exp(u_{kc_k})}{\sum_{j'=1}^{V} \exp(u_{j'})}.$$



Image

Skip-gram model



$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

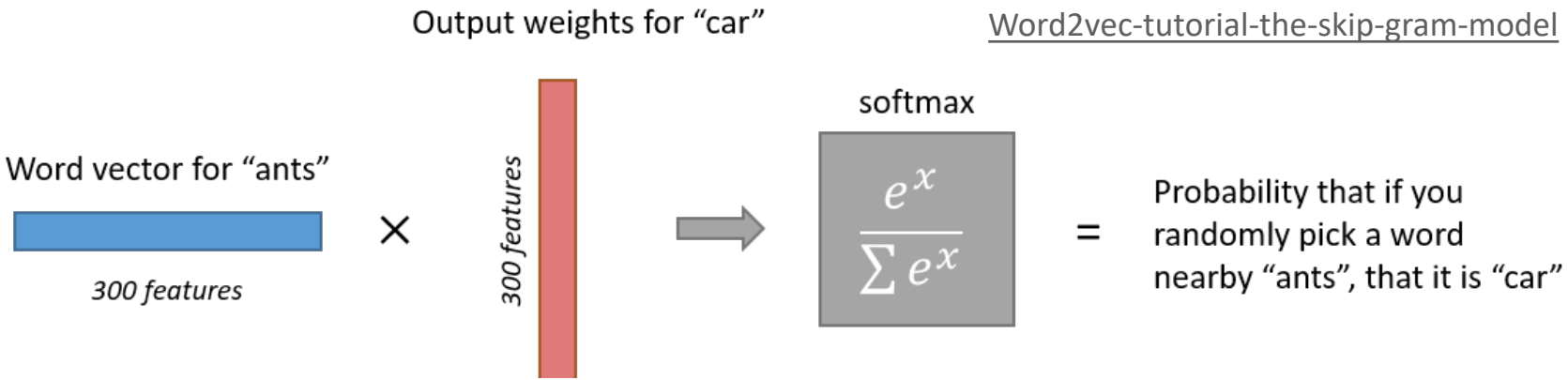
What happens if we multiply one-hot encoding (e.g. k-th element is 1) to matrix $[V \times N]$?

k-th row is selected (and it becomes the value of hidden layer).

Word2vec-tutorial-the-skip-gram-model

Skip-gram model

ITMO UNIVERSITY



$$p(o|i) = \frac{e^{u_i \cdot v_o}}{\sum_K e^{u_i \cdot v_k}}$$

$$L = -\sum_j \ln p(o|i) = -\sum_j \ln \frac{e^{u_i \cdot v_o}}{\sum_K e^{u_i \cdot v_k}}$$

u_i is word vector for input word i
v_o is output weights for output word o
k denotes all words in vocabulary

This formula requires updating for all weights in output layer!

Negative Sampling

ITMO UNIVERSITY

- Negative sampling aims to modify **only a small fraction of weights** for each training example
- Recall that we need to update weight vectors for all of the V words in the dictionary
- Only one of the words is positive (that is, we have one in the output vector)

- Negative sampling: update weight vector for **positive** and **just several negatives**
- **5-20 negative** samples works well for smaller datasets, and **2-5** for large datasets

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

$$p(D=1|i,o) = \sigma(u_i \cdot v_o) = \frac{1}{1+e^{-u_i \cdot v_o}}$$

$$p(D=0|i,k) = 1 - \sigma(u_i \cdot v_o) = \frac{1}{1+e^{u_i \cdot v_o}}$$

$$L = -\log \sigma(u_i \cdot v_o) - \sum_{k \sim P(\omega)} \log \sigma(-u_i \cdot v_k)$$

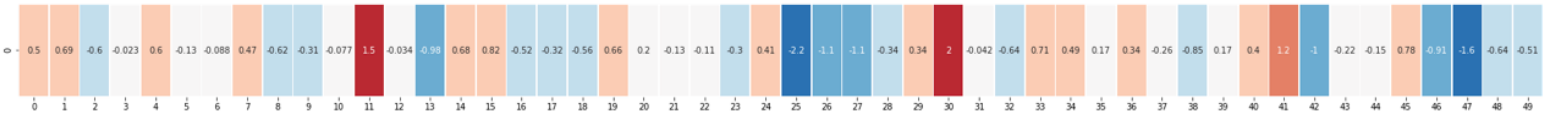
$$P(\omega) \sim U(\omega)^{3/4}$$

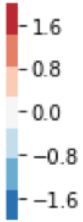
$$P(w_i) = \frac{f(w_i)}{\sum_{j=0}^{n} (f(w_j))}$$

Unigram distribution

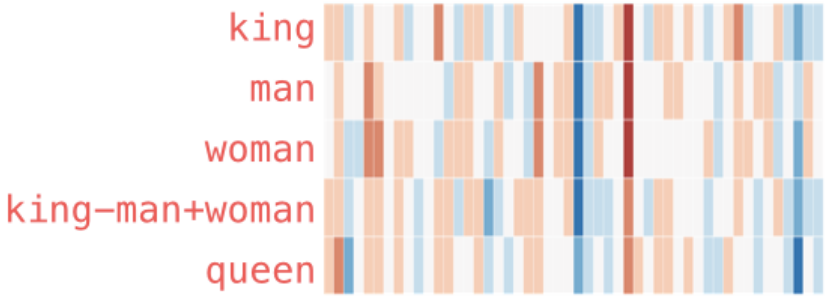
Properties of Word2vec vectors

50-dimensional embeddings pre-trained on the English Wikipedia

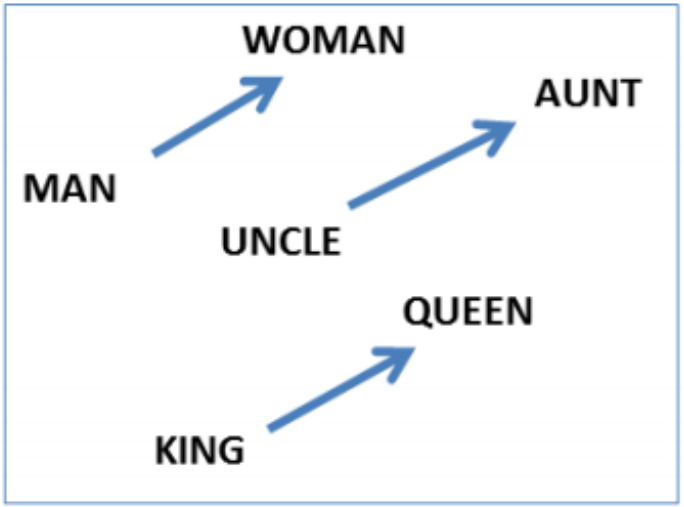


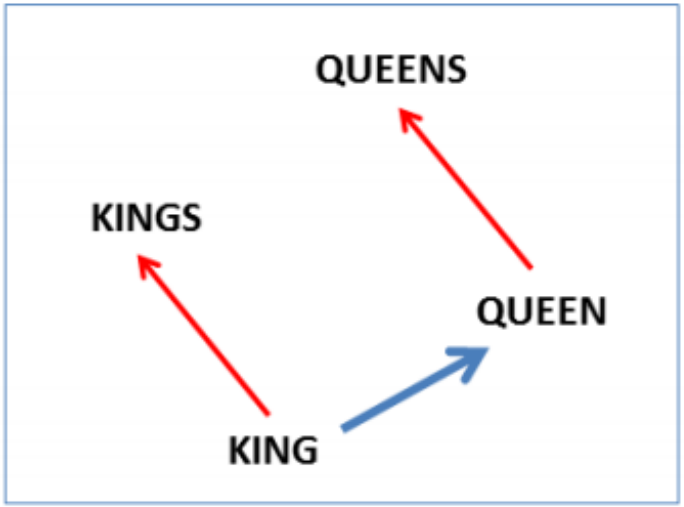


king − man + woman ~= queen



Properties of Word2vec vectors

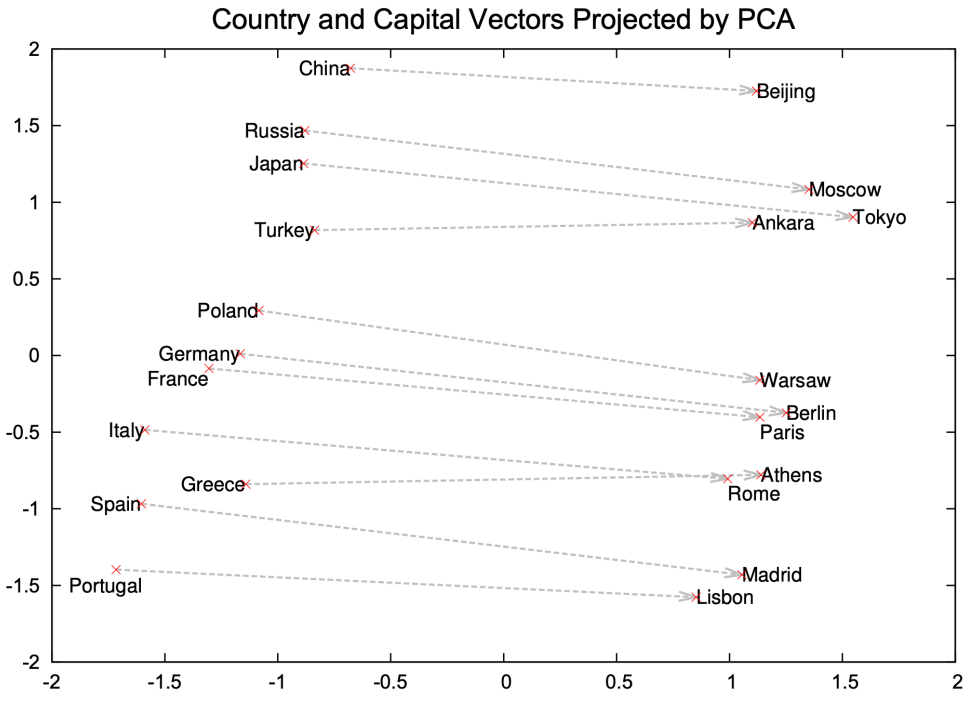




(Mikolov et al., NAACL HLT, 2013)

$$w(Paris) - w(France) + w(Russia) = ?$$

Properties of Word2vec vectors



Mikolov'13

Chris Nicolson on Quora

king::queen::man:
[woman, Attempted abduction, teenager, girl]

house:roof::castle:
[dome,bell_tower, spire, crenellations, turrets]

newyorktimes:sulzberger::fox:
[Murdoch,Chernin, Bancroft, Ailes]

love:indifference::fear:
[apathy, callousness, timidity, helplessness, inaction]

donaldtrump:republican::barackobama:
[Democratic, GOP, Democrats, McCain]

building:architect::software:
[programmer, SecurityCenter, WinPcap]

Main stages:

ITMO UNIVERSITY

- Process raw text data
- Generate training dataset
- Choose on of word2vec model: CBOW/Skip-gram
- Train model
- Get weight matrix

GloVe (2014, Senford)

ITMO UNIVERSITY

GloVe is an unsupervised learning algorithm for obtaining vector representations for words.
Training is performed on aggregated global **word-word co-occurrence statistics** from a corpus.

$$p_{ij} = p(j \mid i) = \frac{X_{ij}}{X_i} = \frac{X_{ij}}{\sum_k X_{ik}}$$

And relation: $\frac{p_{ij}}{p_{kj}}$

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

GloVe

ITMO UNIVERSITY

Collect word co-occurence statistics in a form of word co-ocurrence matrix X.

Cost function $$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij})(w_i^T w_j + b_i + b_j - \log X_{ij})^2$$

$$f(X_{ij}) = \begin{cases} (\frac{X_{ij}}{x_{max}})^{\alpha} & \text{if } X_{ij} < XMAX \\ 1 & \text{otherwise} \end{cases}$$

where w_i - vector for the main word, w_j - vector for the context word, b_i, b_j are scalar biases for the main and context words

GloVe

Advantages

- Simple architecture without neural network.
- The model is fast and this may be sufficient for simple applications.
- Unlike Word2Vec, GloVe takes into account the frequency of occurrence of words.

Disadvantages

- While the co-occurrence matrix provides global information, GloVe remains trained at the word level and provides little data about the sentence and the context in which the word is used.
- Handles unknown and rare words poorly.

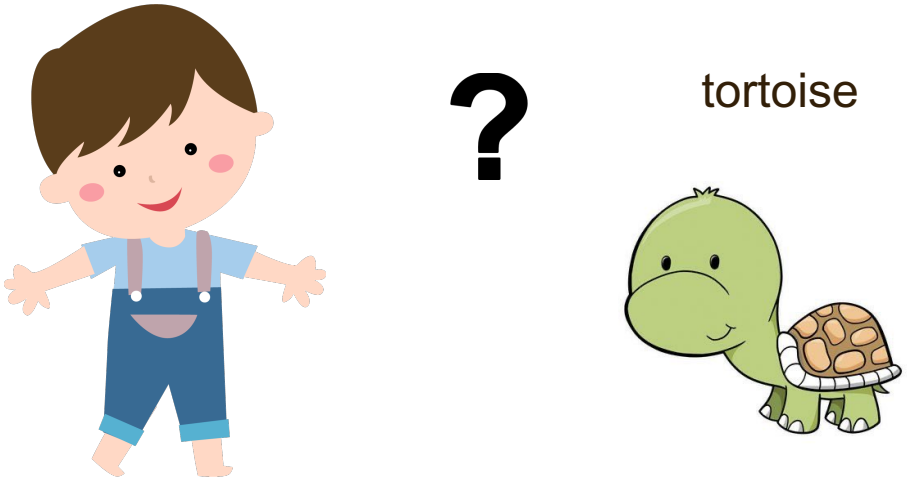
Out-of-vocabulary words

ITMO UNIVERSITY

Out-of-vocabulary (OOV) a problem in computational linguistics and natural language processing when the input data includes words that were not in the dictionary or corpus during training.

Solution:

- deletion
- replacement with a special token



fastText from Faceboook

ITMO UNIVERSITY

- The fastText model builds vector representations of NN-grams.
- The vector representation of a word is the sum of vector representations of all its NN-grams.
- Parts of words are more likely to occur in other words, which makes it possible to produce vector representations for rare words.

Each word w is represented as a bag of character n-gram. We add special boundary symbols < and > at the beginning and end of words, allowing to distinguish prefixes and suffixes from other character sequences. We also include the word w itself in the set of its n-grams, to learn a representation for each word (in addition to character n-grams). Taking the word *where* and $n = 3$ as an example, it will be represented by the character n-grams:

<wh, whe, her, ere, re>

and the special sequence

<where>.

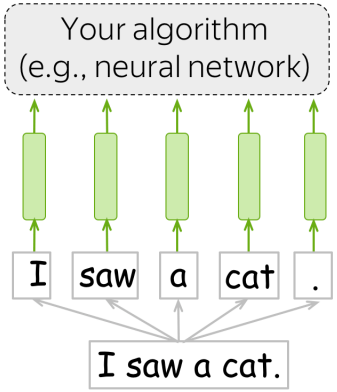
$$s(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^\top \mathbf{v}_c.$$

fastText

Enriching Word Vectors with Subword Information

How to use word vectors?

1. Aggregating vectors for classical classification and clustering algorithms.
   - sum or mean of word representations as sentence/paragraph representation- as a baseline method (Le and Mikolov 2014);
   - a good method for short phrases (Mikolov et al. 2013);
2. Build neural networks on embeddings: for example, LSTM for sentiment analysis

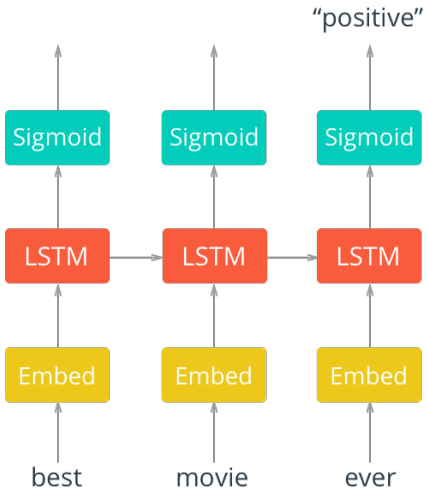


Any algorithm for solving a task

Word representation - vector
(input for your model/algorithm)

Sequence of tokens

Text (your input)



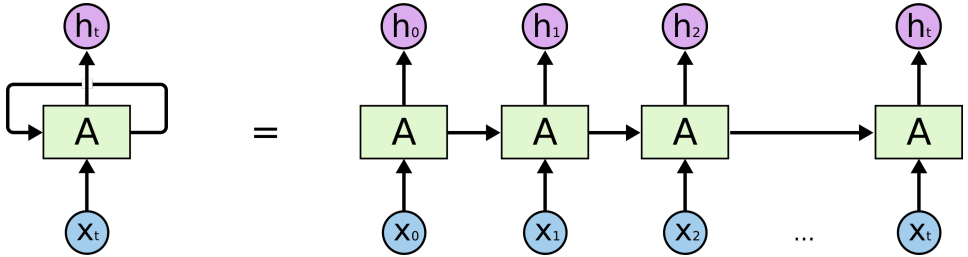
ITMO UNIVERSITY

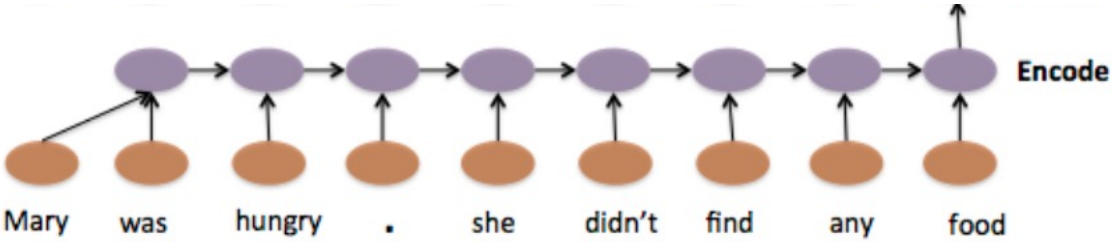
Deep Learning

Recurrent neural network (RNN)

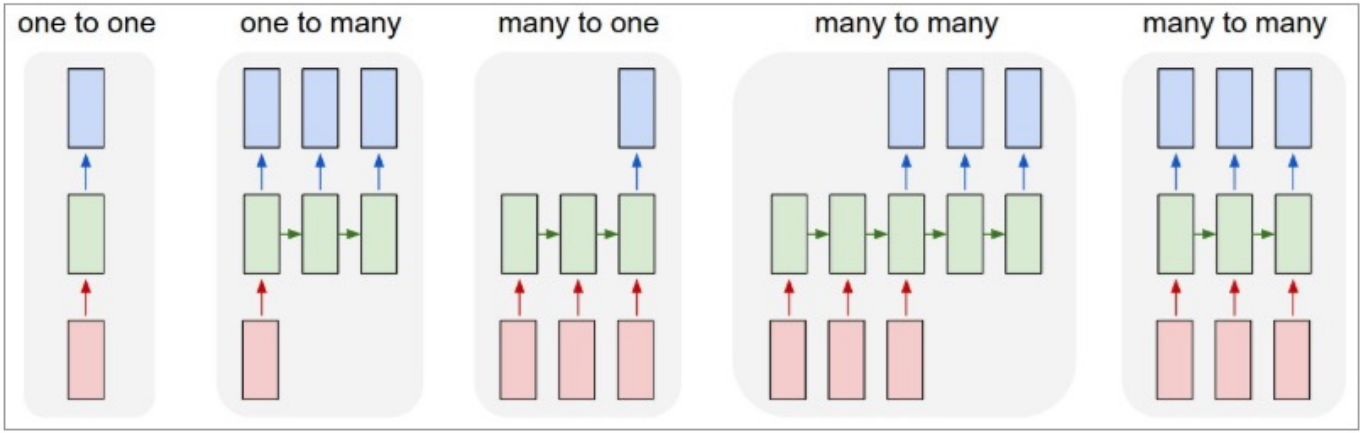
ITMO UNIVERSITY

A recurrent neural network can be thought of as multiple copies of the same node, **each passing a message to a successor**.





Recurrent neural network (RNN)



Classification

Annotation images

Generation text

Text/video/audio classification

Sentiment Analysis

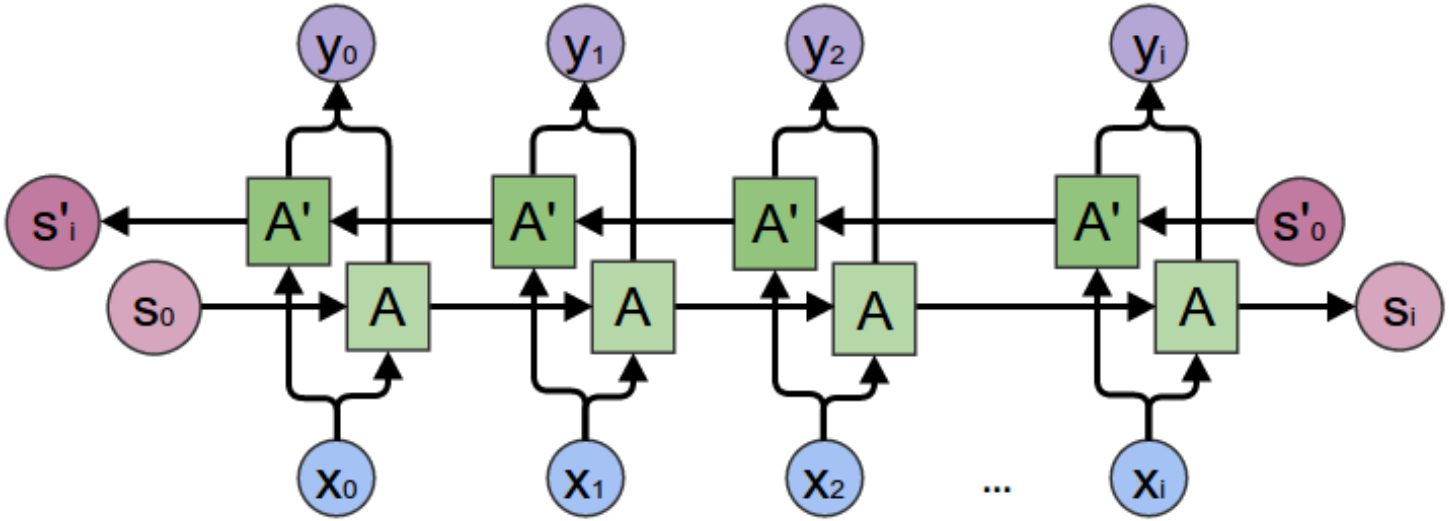
Machine translation

Sequence synchronization

Classification video frames

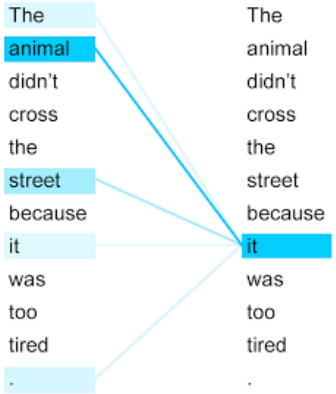
Bidirectional RNN (BRNN)

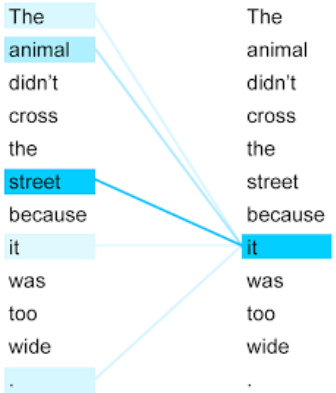
Next words can rule the previous ones:

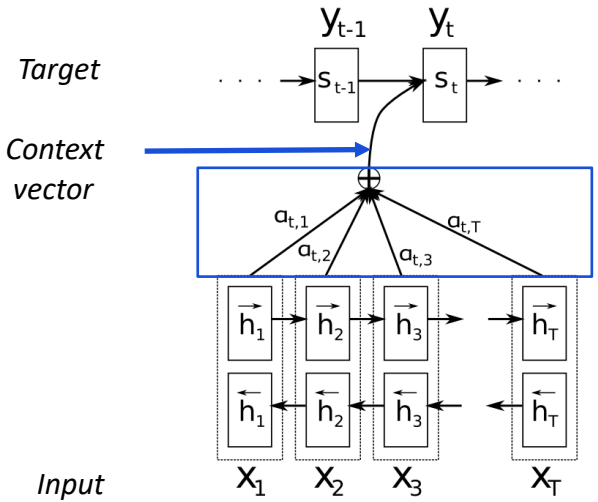


- From beginning to the end →
- From the end to beginning ←
- Learn contexts not only from previous but from next objects (e.g., words);
- Especially important in machine translation;
- In natural language processing, BRNN with LSTM is the most standard model.

State-of-the-art: attention







Decoder: RNN inputs: previous state and dynamic context vector

Attention layer

Encoder: bidirectional RNN

Neural Machine Translation by Jointly Learning to Align and Translate'14

State-of-the-art: transformer

Idea:
each input vector interacts with other words through the attention mechanism

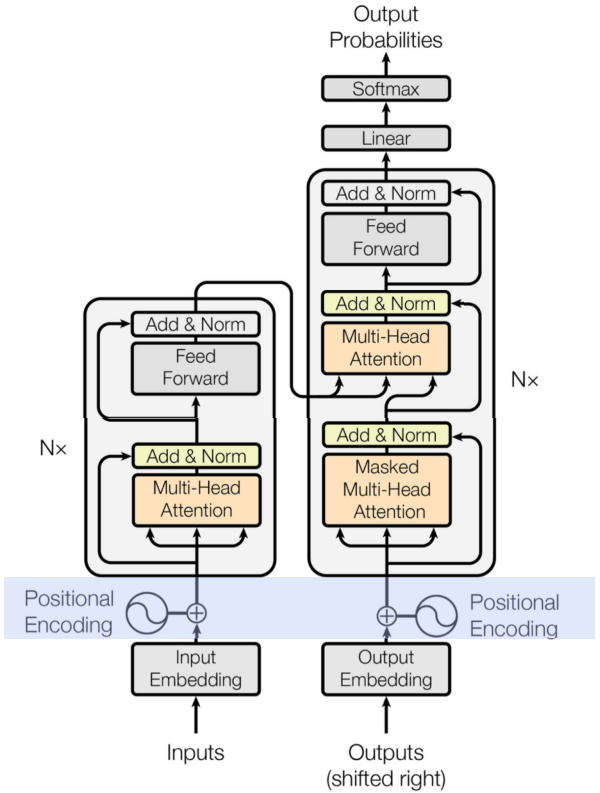


Figure 1: The Transformer - model architecture.

Attention Is All You Need'17

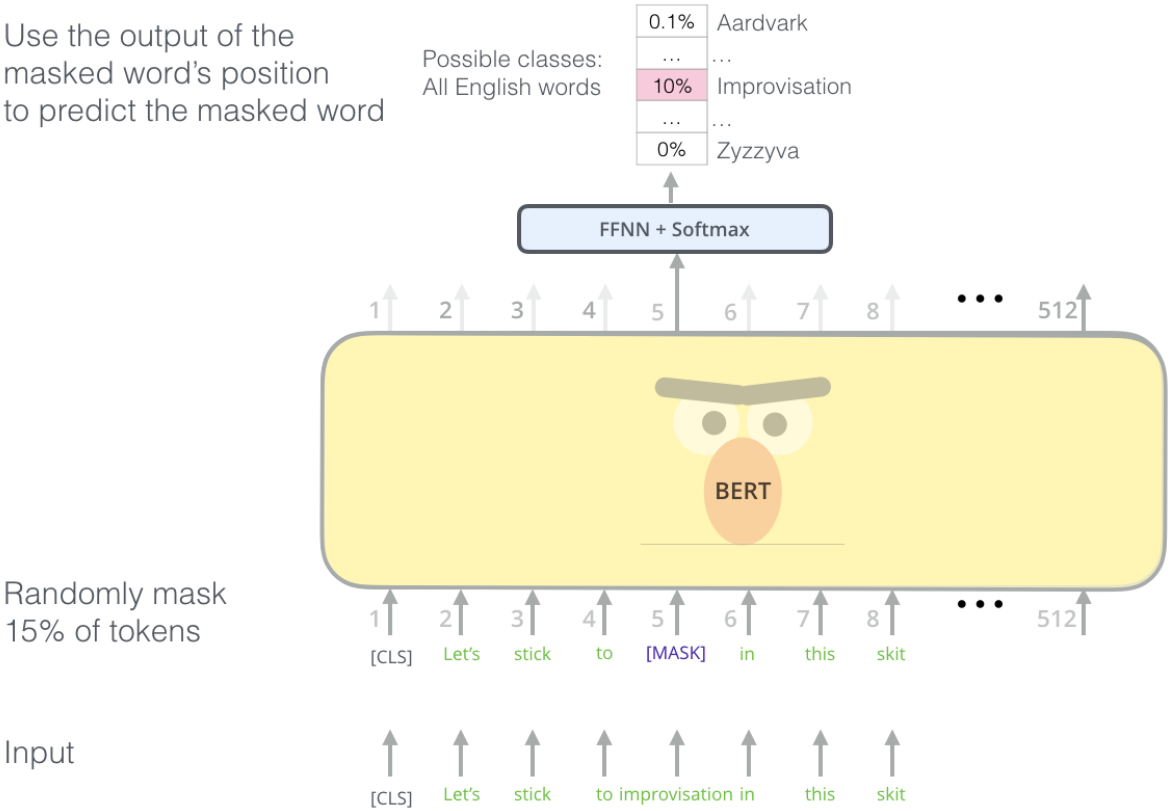
State-of-the-art: BERT

ITMO UNIVERSITY

Language modeling

Tasks:

- Sentences' classification
- Tagging
- Question-answering task
- …



BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding'18

Image source

Summary

ITMO UNIVERSITY

- Text processing tasks are everywhere, but there are a number of complexities involved in their use; this is primarily due to the fact that text data is not structured. To solve most problems, text data must be p**re-processed** (for example, stemming, lemmatization, stopword removal, etc.).
- For specific texts and tasks, non-standard processing may be required.
- The simplest methods for constructing vectors are **BoW and TF-IDF.** The TF-IDF reflects the importance of using each word from a certain set of words in each document.
- **Cosine similarity** is the optimal similarity measure for most representations of natural language texts.
- **Word embeddings** help to represent semantically close words as similar vectors.
- **The OOV** problem can be solved by the fastText model or the creation of an additional token for missed words.
- The RRN can capture dependencies in text data.
- The **attention mechanism** is a technique used in RNN to find relationships between different parts of input and output data.

HW: task 2

ITMO UNIVERSITY

1. Download Alice in Wonderland by Lewis Carroll from Project Gutenberg's website http://www.gutenberg.org/files/11/11-0.txt
2. Perform any necessary preprocessing on the text, including converting to lower case, removing stop words, numbers / non-alphabetic characters, lemmatization.
3. Find Top 10 most important (for example, in terms of TF-IDF metric) words from each chapter in the text (not "Alice"); how would you name each chapter according to the identified tokens?
4. Find the Top 10 most used verbs in sentences with Alice. What does Alice do most often?

ITMO UNIVERSITY

Saint Petersburg, Russia

Thank you for attention!

Severiukhina Oksana
oseveryukhina@gmail.com