NLP.Word Embeddings

MIPT
17.02.2022
Anton Emelianov, Albina Akhmetgareeva

https://github.com/king-menin/mipt-nlp2022

Today

- What is word representations?
  Why do we need them?
- One-hot Vectors
- Distributional Semantics
- Count-Based Methods
- Prediction-based Method
  - WORD2VEC
- Evaluation of Word embeddings

Word representations

What is word representations?

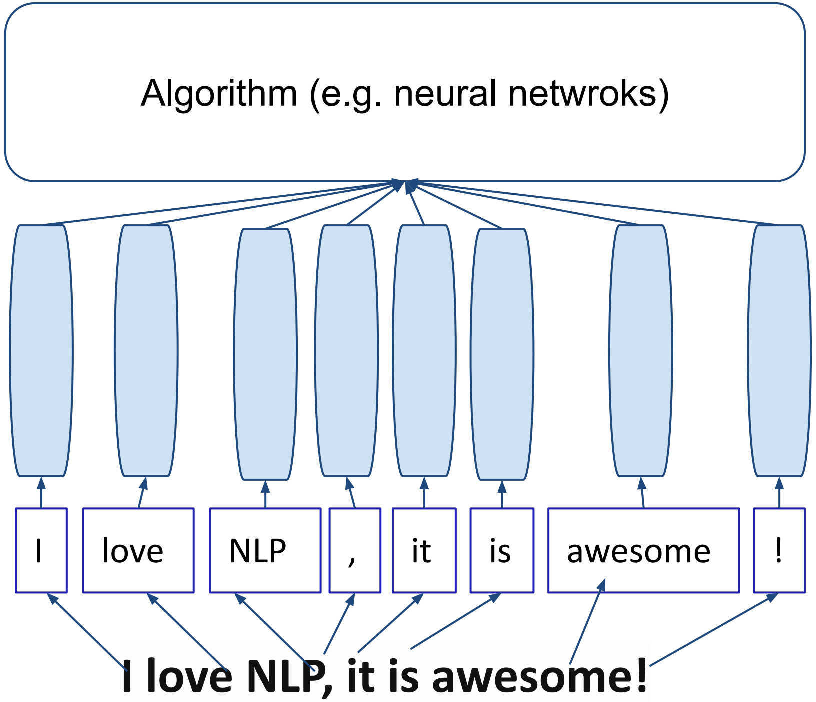
Why do we need them?

- Machine learning models “see” data differently from how we (humans) do.
- Different types of data. What to do with **text**?

For example, sequence:

I love NLP, it’s awesome!

Word representations



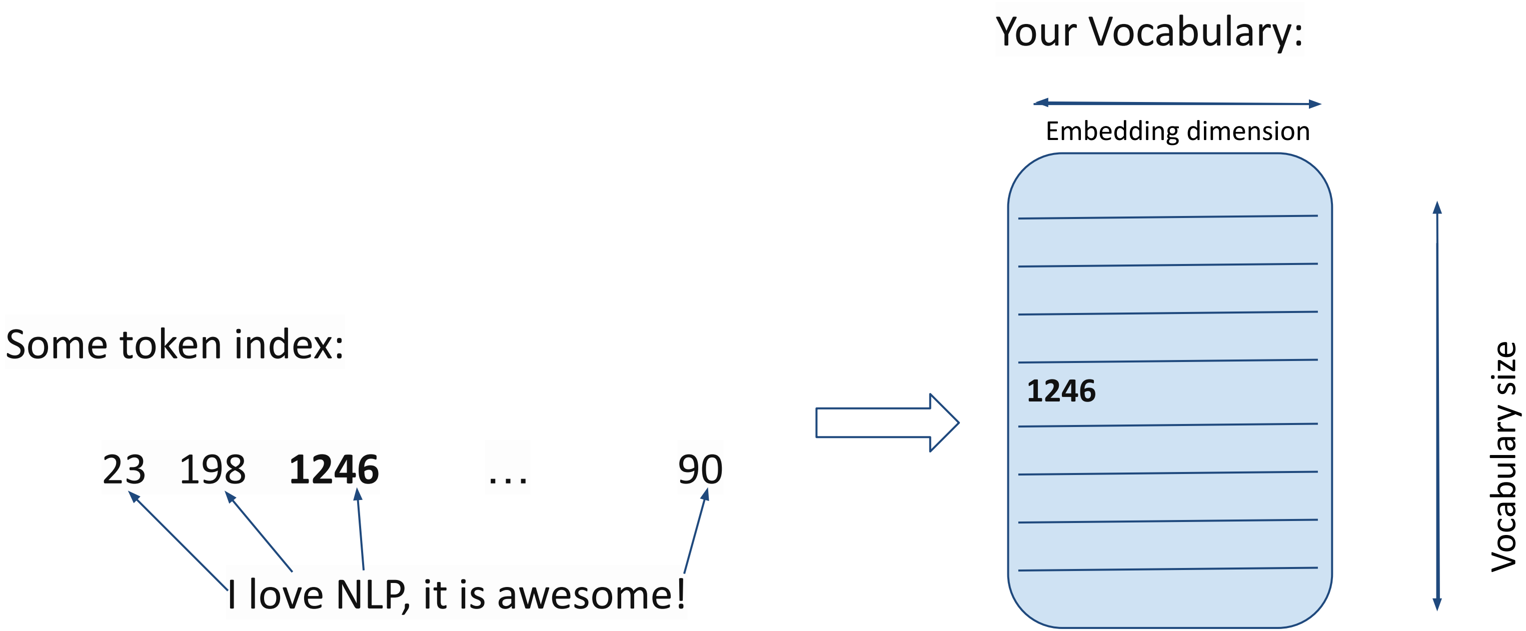
Any algorithm you want
to solve your task

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

Sequence of tokens

Input text

Look-up Table



- For each vocabulary word, a look-up table contains its embedding
- Vocabulary is fixed
- Unknown words => special token or zeros

One hot encoding

- Represent Words as Discrete Symbols
- Enumerate all words in the Vocabulary
- For the i-th word in the vocabulary we get the vector that has 1 on the i-th dimension and 0 on the rest

hotel	[0 0 0 0 0 0 1 0 0 0 0]
hostel	[0 0 1 0 0 0 0 0 0 0 0]

One hot encoding

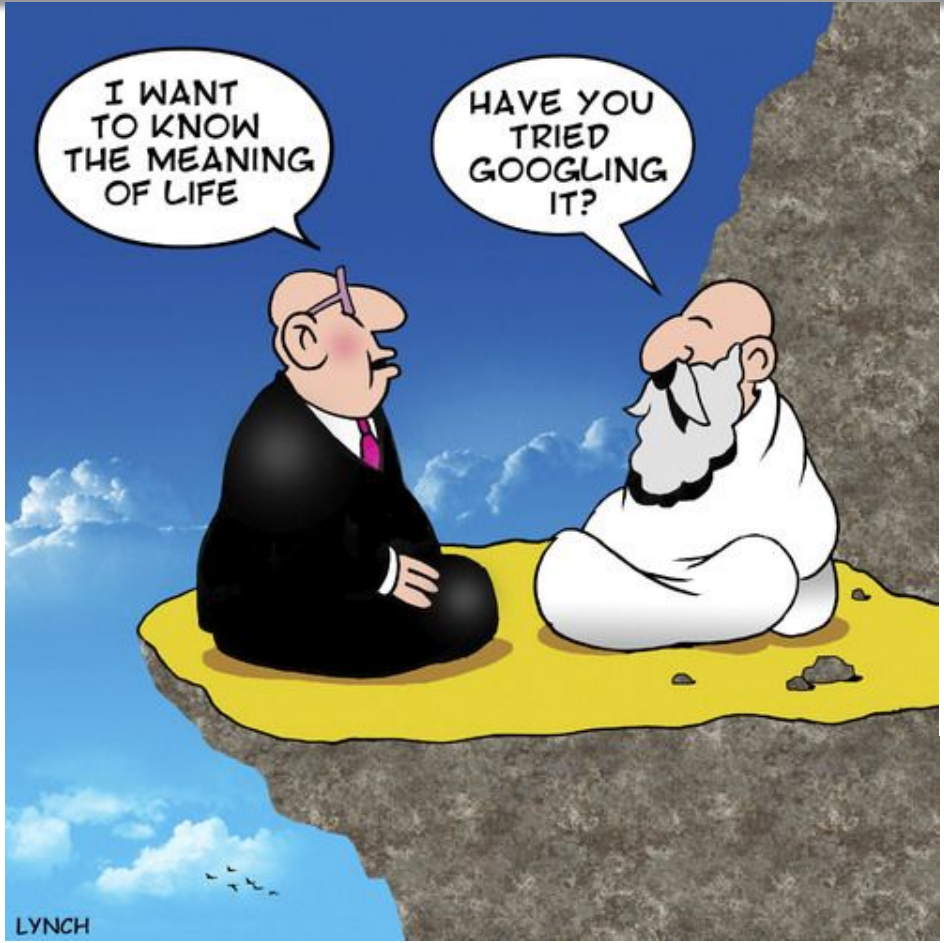
We want to find in Booking not just Hotels in Barcelona but we also consider hostels and AirBNB:

hotel	[0 0 0 0 0 0 1 0 0 0 0]
hostel	[0 0 1 0 0 0 0 0 0 0 0]

Fail:

- Vector size is too large
- These vectors are ortogonal
- They know NOTHING about the words:
  - similarity between words
  - **meaning**

What is meaning?



Distributional semantics

Define MEANING

but first….

What is **Троллингер (Trollinger)?**

Distributional semantics

If you add Троллингер in contexts:

1) Не садись за руль после ______
2) В Италии ___ известен под именами вернач и скьява
3) Бутылка ____ стояла на столе
4) Сорт винограда ____ а также одноименная марка вина выращивается в немецком регионе Вюртемберг

Distributional semantics

1) Не садись за руль после ______
2) В Италии ___ известен под именами вернач и скьява
3) Бутылка ____ стояла на столе
4) Сорт винограда ____ а также одноименная марка вина выращивается в немецком регионе Вюртемберг

Троллингер
(trollinger)
=
a sort of wine
!=
Troll

	(1)	(2)	(3)	(4)	...
троллингер	1	1	1	1	
масло	0	0	1	0	
вино	1	0	1	0	
пятница	0	0	0	0	

Distributional hypothesis

Words which frequently appear in similar contexts have similar meaning.

(Harris 1954, Firth 1957)

We need to put information about **word contexts** into word representation.

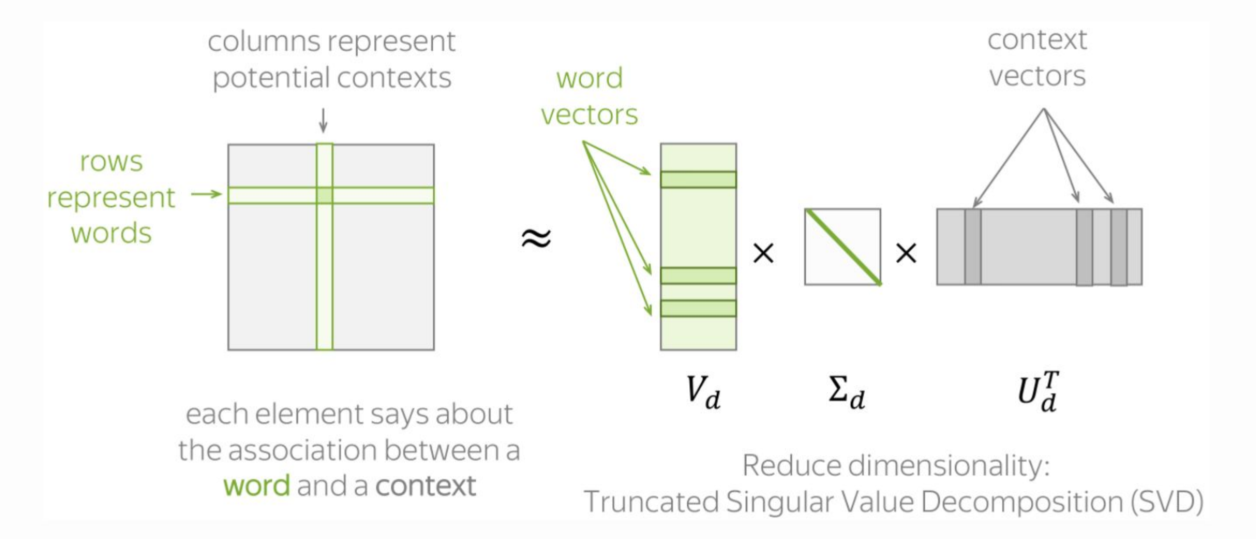
Distributional hypothesis

подтягивают. Корректируя детали, но сохраняя	**вектор**	. Простая штука, однако. Отдельно.
Богуславский — всемерно поддерживали главный гуманистический	**вектор**	педагогических устремлений Сухомлинского.
более доступной цене. «Мы определяем	**вектор**	перемен» Сергей Кадочников, директор НИУ
далее. Музей подсказывает детям правильный	**вектор**	развития. 2. Взрослым.
тоже считаем, что задаем правильный	**вектор**	развития для детей и главной
командами, в существенной мере определяющими	**вектор**	социально-экономического развития страны.
	Вектор	развития биологии, да и всего
экономическом форуме доклад «Россия: восточный	**вектор**	», заявляет, что новая волна освоения
не расплыться повествованию — четкий временной	**вектор**	, хронологическая последовательность: жизнь Чудакова прослеживается
это глобальный геополитический и макроэкономический	**вектор**	развития государства.
ограничивается поддержка фонда, — задается особый	**вектор**	замедления времени, похожий на звук

We will build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts.

Word vectors are also called **word embeddings** or (neural) **word representations**

Count-based methods



HOW?

(1) define what is context

(2) how to compute matrix elements

Two steps:

(1) construct a word-context matrix for our vocabulary

($M[i,j]$ =$f(wi,cj)$ dimension $|Vc|\times|Vw|$, where the element $f(wi, cj)$ will describe the relationship of the word wi with the context cj.)

(2) reduce its dimensionality (SVD)

Count-based methods

1. **Co-Occurrence Counts**

   Context: set L-sized context window

   Matrix element:

   N(w, c) – number of times word w appears in context c

2-sized window

[I love **NLP**, it] is awesome!

Count-based methods

2. PPMI (positive Pointwise Mutual Information)

Context: L-sized context window

Matrix element:

$PPMI(w, c)$ = max(0, PMI(w, c)), where

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

Count-based methods

3. Context:

- document d (from a collection D)

Matrix element:

- tf-idf(w, d, D) = tf(w, d) · idf(w, D)

$$\mathrm{tf}(w, d) = N(w, d) \qquad \mathrm{idf}(w, D) = \log \frac{|D|}{|\{d \in D : w \in d\}|}$$

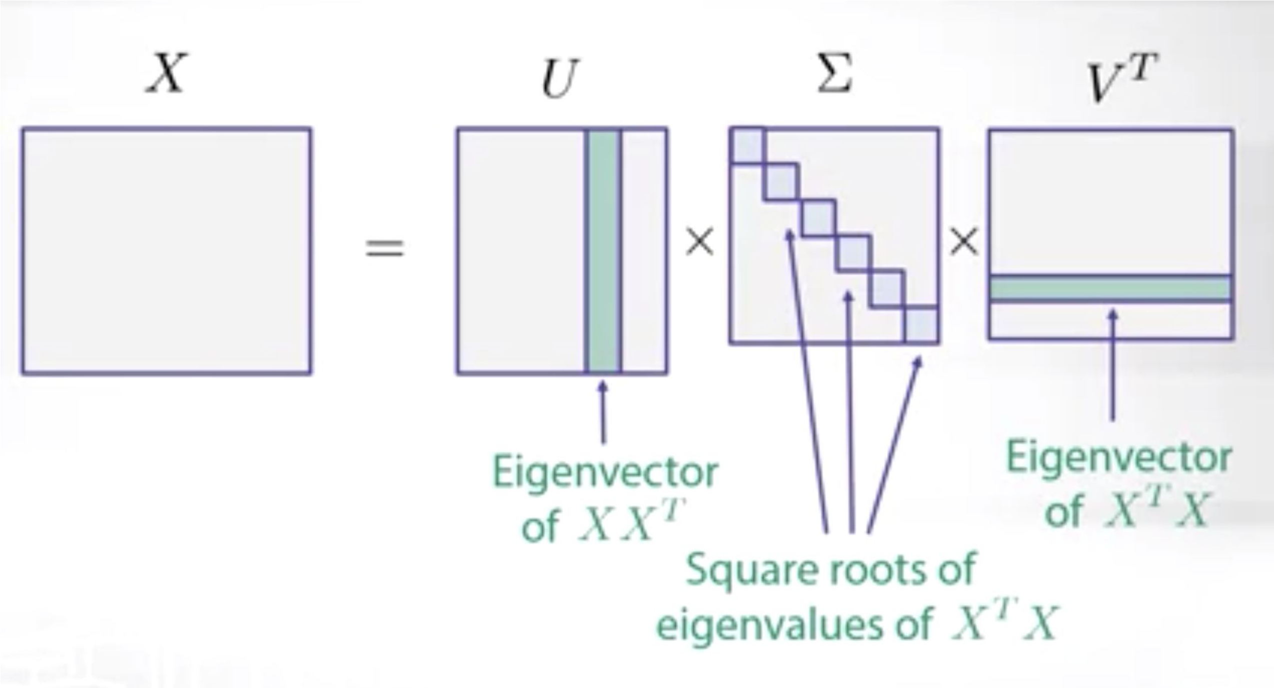
term frequency

inverse document frequency

Dimensionality reduction

Singular value decomposition (SVD) of word-context matrix

Make word vector of new dimension $N \ll |Vc|$

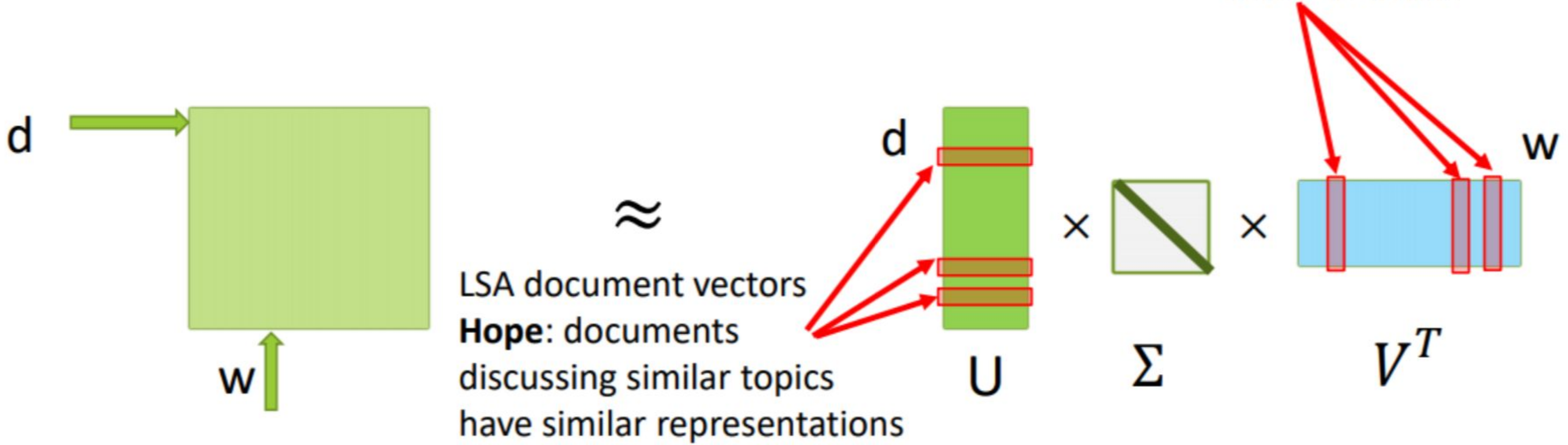


LSA

X - document-term co-occurrence matrix

$$X \approx \hat{X} = U \Sigma V^T$$

LSA term vectors
Hope: term having common meaning are mapped to the same direction

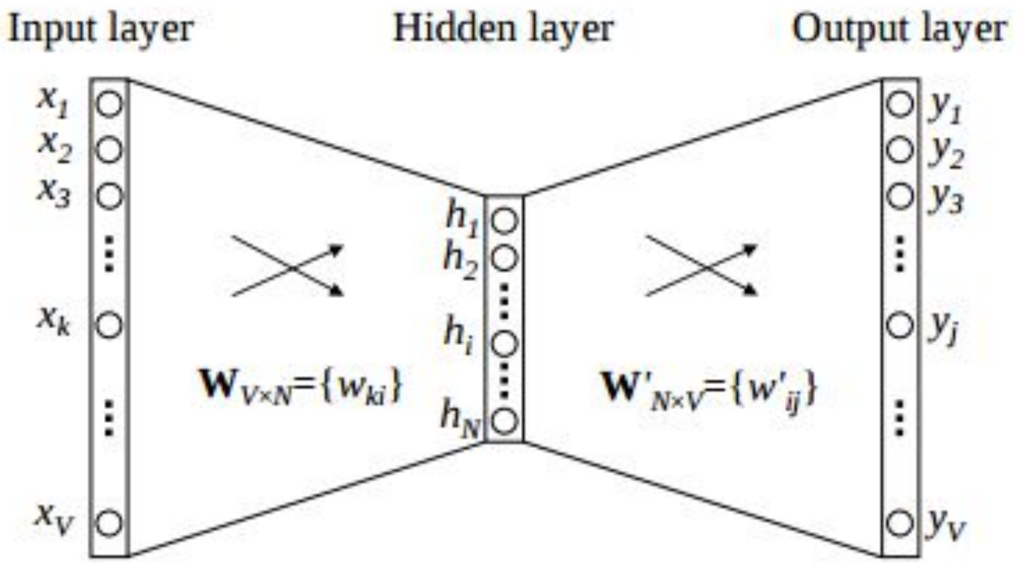


Predicted-based. Word2Vec

T. Mikolov, K. Chen, G. Corrado, J. Dean. Efficient Estimation of Word Representations in Vector Space (2013).
T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean. Distributed Representations of Words and Phrases and their Compositionality (2013).

Word2Vec

Learn word vectors by training them to predict contexts!

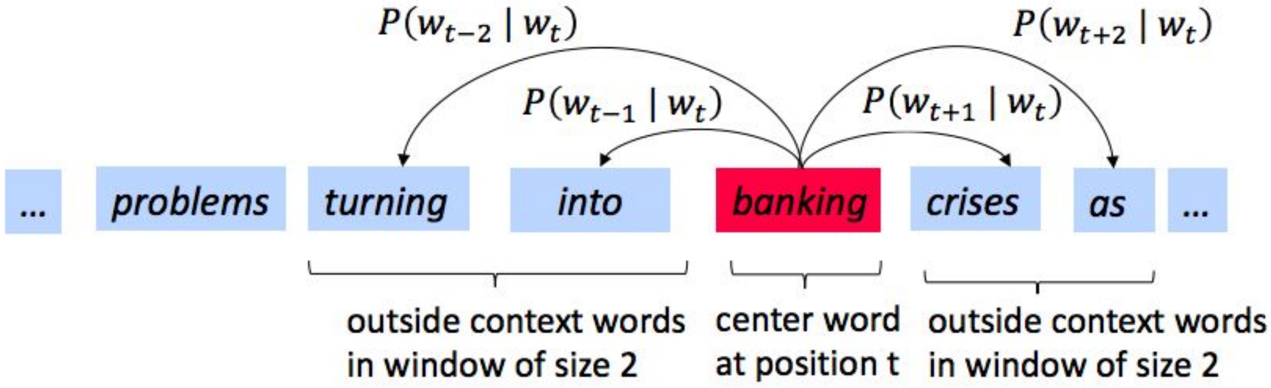


word2vec model architecture

Word2Vec

General pipeline:

- take a huge text corpus;
- go over the text with a sliding window, moving one word at a time;
- for the focus word, compute probabilities of context words;
- adjust the vectors to increase these probabilities.



Word2Vec. Objective function

For each position $t = 1, \dots, T$, predict context words within a window of fixed size *m*, given center word wj. Data likelihood:

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \ne 0}} P(w_{t+j} \mid w_t; \theta)$$

θ is all variables to be optimized

The objective function (aka loss function or cost function) J(θ) is the average negative log-likelihood:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \ne 0}} \log P(w_{t+j} \mid w_t; \theta)$$

Minimizing objective function ⟺ Maximizing predictive accuracy

Word2Vec

- **Question:** How to calculate $P(w_{t+j}|w_j, \theta)$?
- **Answer:** We will *use two* vectors per word *w*:

  v- when *w* is a center word

  u- when *w* is a context word

- Then for a center word *c* and a context word *o*:

$$P(o|c) = \frac{\exp(\boxed{u_o^T v_c})}{\sum_{w \in V} \exp(u_w^T v_c)}$$

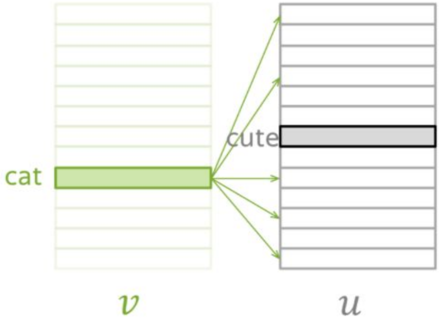
Dot product: measures similarity of *o* and *c*
Larger dot product = larger probability

Normalize over entire vocabulary
to get probability distribution

$$\Rightarrow \quad softmax(x_i) = \frac{\exp(x_i)}{\sum_{j=i}^{n} \exp(x_j)}.$$

Word2Vec. Training steps

1. Take dot product of v_{cat} with **all** u 2. exp 3. sum all



$$u_{w1}^T v_{cat} \longrightarrow \exp(u_{w1}^T v_{cat})$$
$$u_{w3}^T v_{cat} \longrightarrow \exp(u_{w3}^T v_{cat})$$
$$\vdots$$
$$\boxed{u_{cute}^T v_{cat}} \longrightarrow \exp(u_{cute}^T v_{cat})$$
$$\vdots \quad 1$$
$$u_{wn}^T v_{cat} \longrightarrow \exp(u_{wn}^T v_{cat})$$

$$\boxed{\sum_{w \in V} \exp(u_w^T v_{cat})}$$

2

4. get loss (for this one step)

$$J_{t,j}(\theta) = \underbrace{-u_{cute}^T v_{cat}}_{1} + \underbrace{\log \sum_{w \in V} \exp(u_w^T v_{cat})}_{2}$$

5. evaluate the gradient, make an update

$$v_{cat} := v_{cat} - \alpha \frac{\partial J_{t,j}(\theta)}{\partial v_{cat}}$$

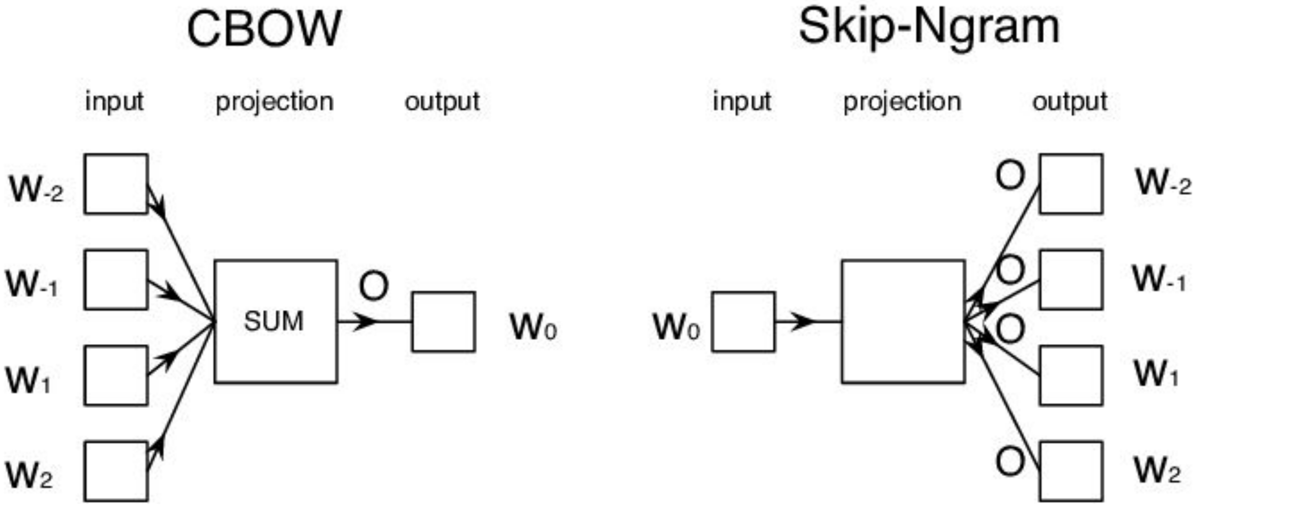
$$u_w := u_w - \alpha \frac{\partial J_{t,j}(\theta)}{\partial u_w} \ \forall\, w \in \mathrm{V}$$

Word2Vec

Why two vectors? Easier optimization. Average both at the end.

Two model variants:

1. Skip-grams (SG) Predict context ("outside") words (position independent) given center word
2. Continuous Bag of Words (CBOW) Predict center word from (bag of) context words



Word2Vec. Negative sampling

Parameters to be updated: • for all the vocabulary |V| **BAD and TIME-CONSUMING**

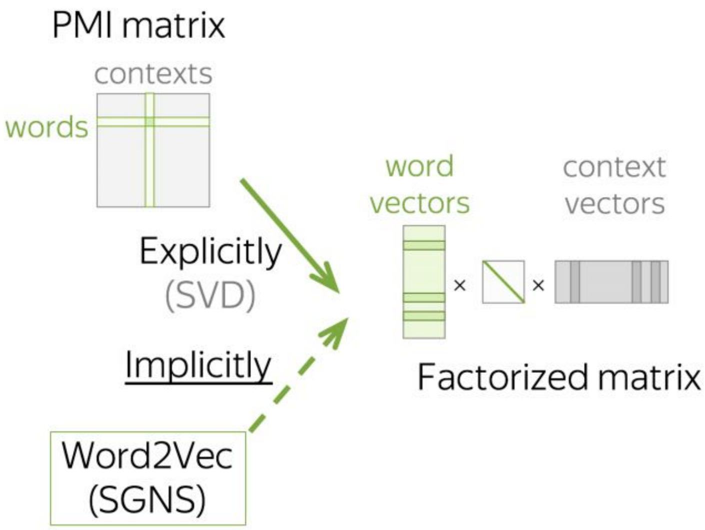
Let’s take Negative Samples: randomly selected K words. Updated for them! **FASTER**

How to choose negative?

randomly samples negative examples based on the empirical distribution of words.

Word2Vec + Skip-Gram with Negative Sampling

implicitly approximates the factorization of a (shifted) PMI matrix



Word2Vec. Hyperparameters

- Model: Skip-Gram with negative sampling;
- Number of negative examples: for smaller datasets, 15-20; for huge datasets (which are usually used) it can be 2-5;
- Embedding dimensionality: frequently used value is 300, but other variants (e.g., 100 or 50) are also possible;
- Sliding window (context) size: 5-10:
  - Larger windows – more topical similarities
  - Smaller windows – more functional and syntactic similarities

GLOVE

Mix of Count-based and Prediction-based methods:

- count global corpus statistics
- learn vectors

Controls the influence of rare and frequent words: loss for each pair (w, c) is weighted in a way that

- rare events are penalized,
- very frequent events are not over-weighted.

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

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

So many embeddings….

- Skip-gram, CBOW aka word2vec [Mikolov et al., 2013] – https://www.tensorflow.org/tutorials/word2vec
- Dependency embeddings [Levi et al., 2015] – https://bitbucket.org/yoavgo/word2vecf
- GloVe [Pennington et al., 2014] – https://nlp.stanford.edu/projects/glove/
- FastText [Joulin et al., 2016] – https://github.com/facebookresearch/fastText
- AdaGram [Bartunov et al., 2016] – https://github.com/sbos/AdaGram.jl – http://adagram.ll-cl.org
- SenseGram [Pelevina et al., 2016] – https://github.com/tudarmstadt-lt/sensegram
- StarSpace [Wu, 2017] – https://github.com/facebookresearch/StarSpace
- Poincare embeddings [Nickel et al., 2017]
- Doc2Vec [Quoc Le, Mikolov, 2014] - https://arxiv.org/pdf/1405.4053v2.pdf

RUSVECTORES

http://rusvectores.org/ru/

Words similarity

- **Cosine similarity**

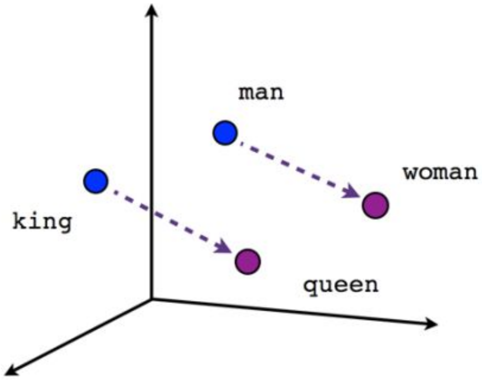
$$\cos(u,v) = \frac{uv}{\|u\|_2 \|v\|_2} = \frac{\sum_i u_i v_i}{\sqrt{\sum_i {u_i}^2}\sqrt{\sum_i {v_i}^2}}$$

- **Jaccard index**, also known as the **Jaccard similarity coefficient**

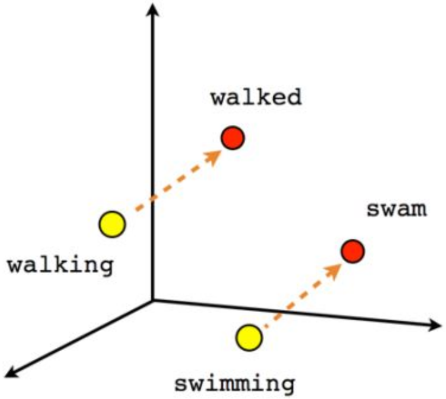
$$jc(u,i) = \frac{\sum_i \min(u_i, v_i)}{\sum_i \max(u_i, v_i)}$$

Linear structure

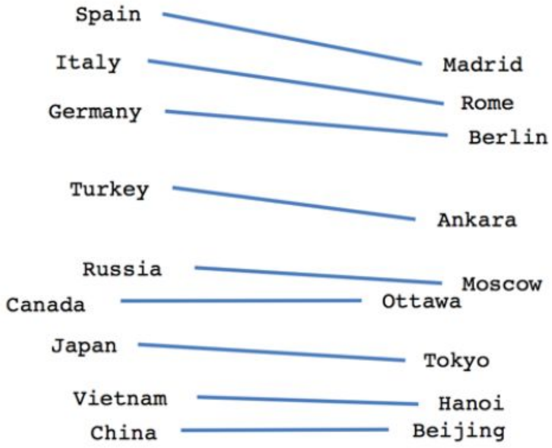
Semantic and syntastic relationships are linear in word vector space



Male-Female



Verb tense



Country-Capital

v(king) - v(man) + v(woman) = v(queen)

v(big) - v(small) + v(smallest) = v(biggest)

Evaluation

Intrinsic:

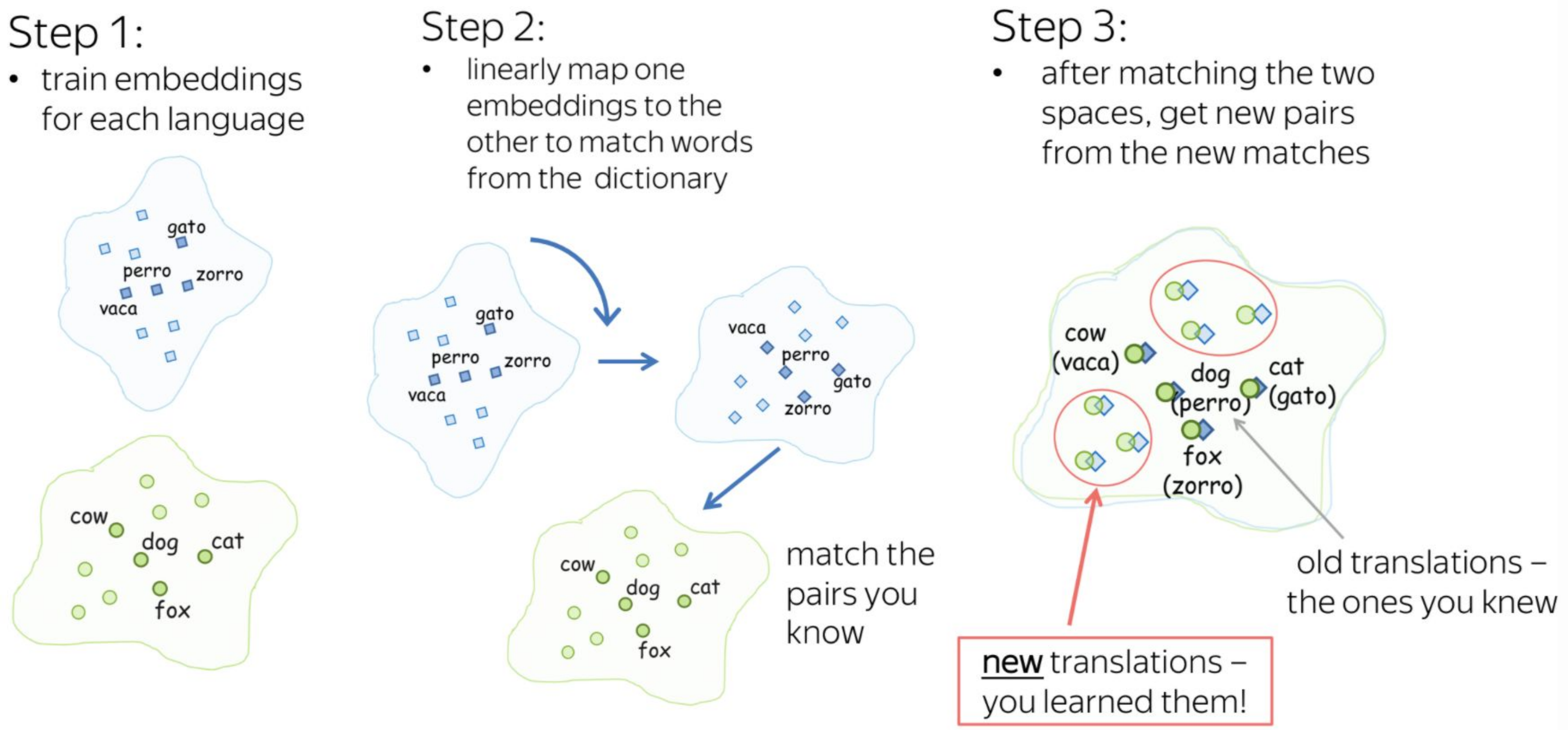
- **Evaluation on a specific/intermediate subtask**
  Fast to compute
  Helps to understand system
  Needs positive correlation with real task to determine usefulness

Extrinsic:

- **Evaluation on a real task**
  Expensive (maybe long)
  Need to train the same model several times: one model for each embedding set

Vectors of multiple languages

- Corpus in one language and corpus in another. Not parallel!
- Can build new dictionaries



Diachrony

Detect Words that Changed Their Usage

Shiftry

https://shiftry.rusvectores.org/ru/

- web service for analyzing diachronic changes in the usage of words occurring in news texts from Russian mass media
- explore the semantic shifts history of any given query word
- visualizations of the words' trajectories through time.

Questions

References

Reference lists

- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov, Enriching word vectors with subword information, arXiv preprint arXiv:1607.04606 (2016).
- Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, and Dmitry Vetrov, Breaking sticks and ambiguities with adaptive skip-gram, Artificial Intelligence and Statistics, 2016, pp. 130–138.
- Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Herv´e J´egou, Word translation without parallel data, arXiv preprint arXiv:1710.04087 (2017).
- William L Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky, Inducing domain-specific sentiment lexicons from unlabeled corpora, Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing, vol. 2016, NIH Public Access, 2016, p. 595.
- William L Hamilton, Jure Leskovec, and Dan Jurafsky, Diachronic word embeddings reveal statistical laws of semantic change, arXiv preprint arXiv:1605.09096 (2016).
- Andrey Kutuzov and Elizaveta Kuzmenko, Webvectors: a toolkit for building web interfaces for vector semantic models, International Conference on Analysis of Images, Social Networks and Texts, Springer, 2016, pp. 155–16
- Omer Levy and Yoav Goldberg, Dependency-based word embeddings, Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), vol. 2, 2014, pp. 302– 308. 1.
- Quoc Le and Tomas Mikolov, Distributed representations of sentences and documents, International Conference on Machine Learning, 2014, pp. 1188–1196.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781 (2013).
- Jeffrey Pennington, Richard Socher, and Christopher Manning, Glove: Global vectors for word representation, Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.
- Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims, Evaluation methods for unsupervised word embeddings, Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 298–307.
- Joseph Turian, Lev Ratinov, and Yoshua Bengio, Word representations: a simple and general method for semi-supervised learning, Proceedings of the 48th annual meeting of the association for computational linguistics, Association for Computational Linguistics, 2010, pp. 384–394.
- Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning, Bilingual word embeddings for phrase-based machine translation, Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, pp. 1393–1398.