Lecture 9:
Introduction to texts

Ivan Provilkov

Plan

- NLP tasks
- Text classification overview
- Word representations
- LSA
- TFIDF

Based on
https://github.com/yandexdataschool/nlp_course & https://github.com/neychev/harbour_dlia2019

Natural Language Processing (NLP)

This is a subfield of computer science about how to program computers to deal with natural languages (English, Russian...).

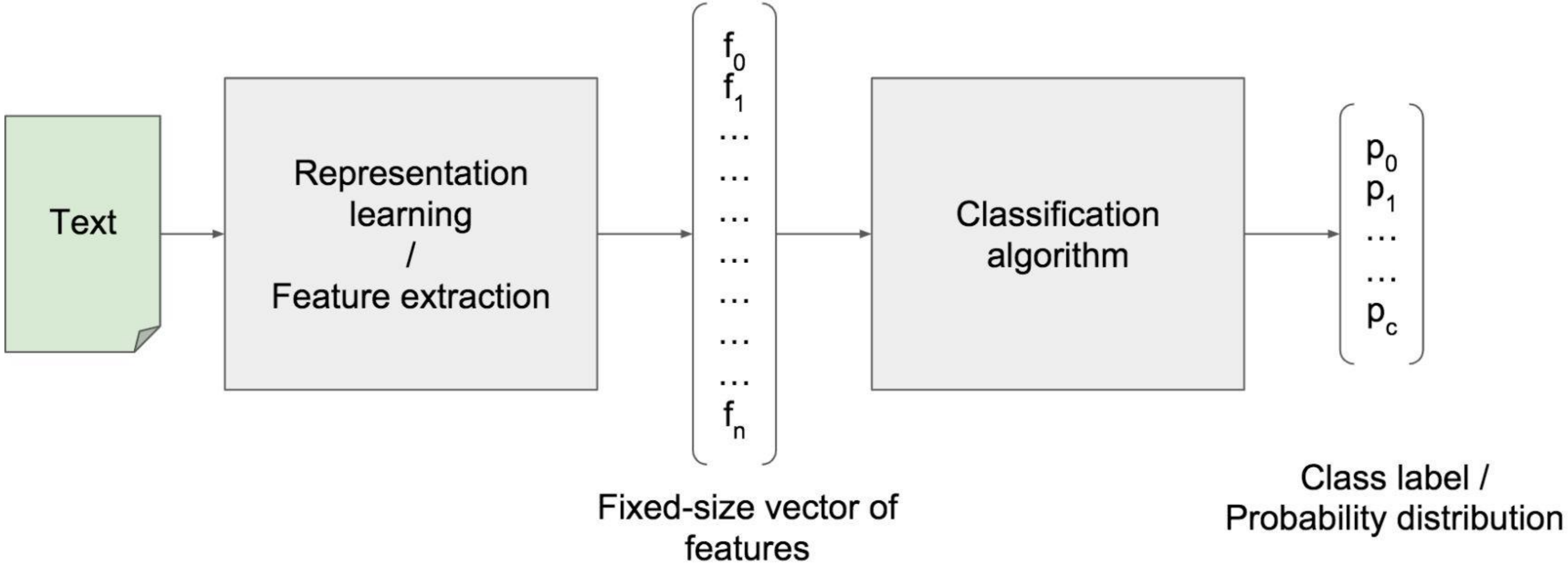
NLP tasks (1)

- Machine Translation
  Between different languages
- Language modeling
  Model which can predict probability of sentence, or word, given context
- Part of speech tagging
  Determine part of speech for each word
- Parsing
  Determine parse tree for sentence which shows relations between words

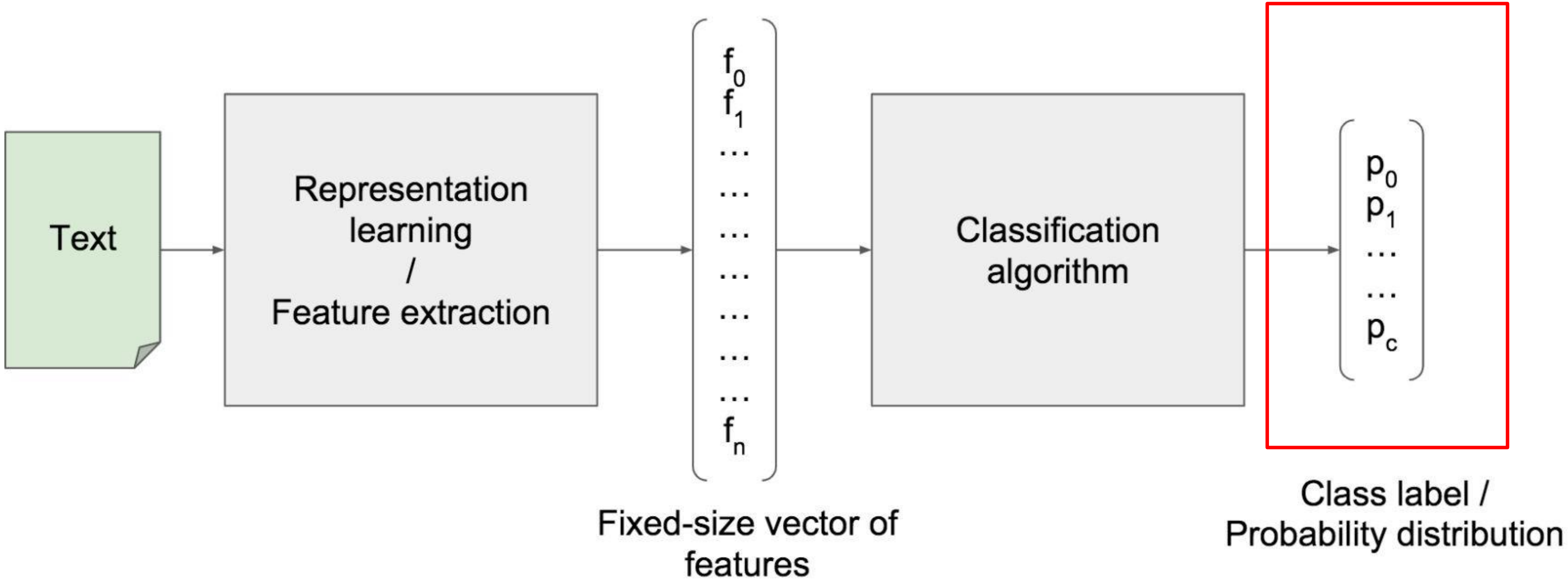
NLP tasks (2)

- Natural language generation
  Convert some information (images, digits) into human readable way
- Named entity recognition
  Determine which items in text map to proper classes. For example, people or organizations
- Question answering
  Given a human language question, determine its answer
- Topic modeling
  Extract topics and determine which relate to text
- ...

Text classification



Text classification



Text labels

- Discrete labels:
  - Binary: spam filtering, sentiment analysis

Text labels

- Discrete labels:
  - Binary: spam filtering, sentiment analysis
  - Multi-class: categorization of items by its description

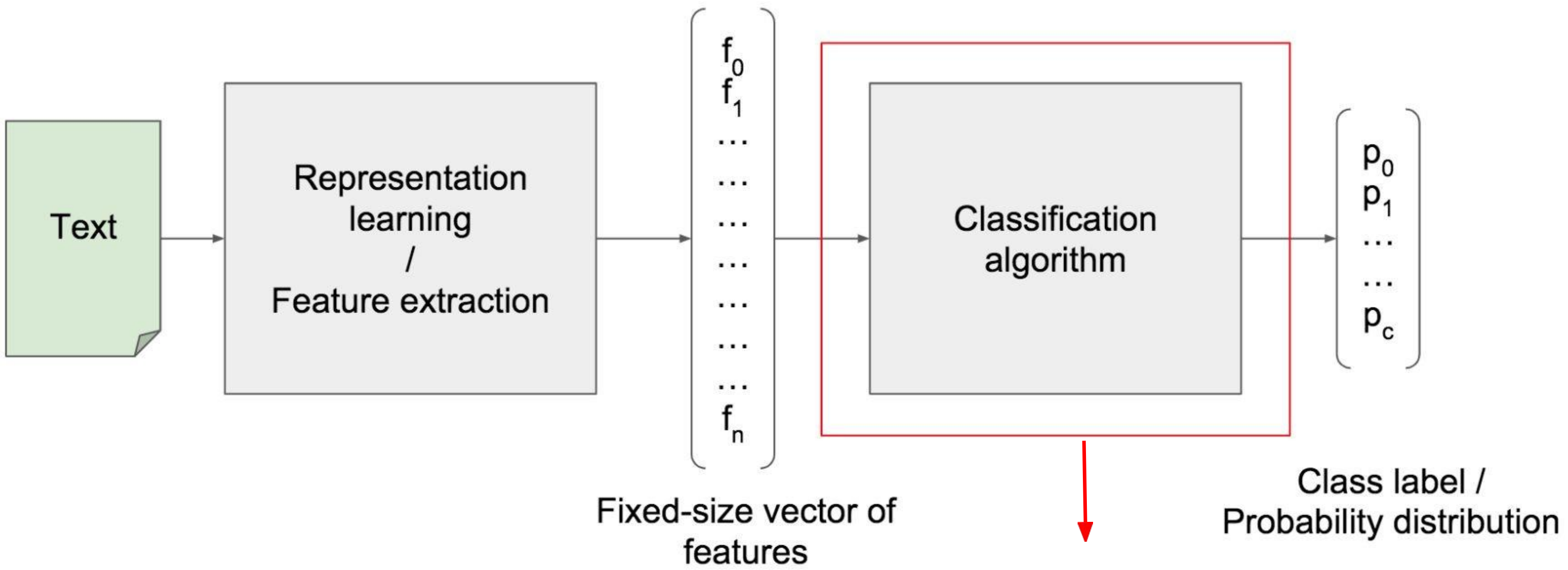
Text labels

- Discrete labels:
  - Binary: spam filtering, sentiment analysis
  - Multi-class: categorization of items by its description
  - Multi-label: #hashtag prediction

Text labels

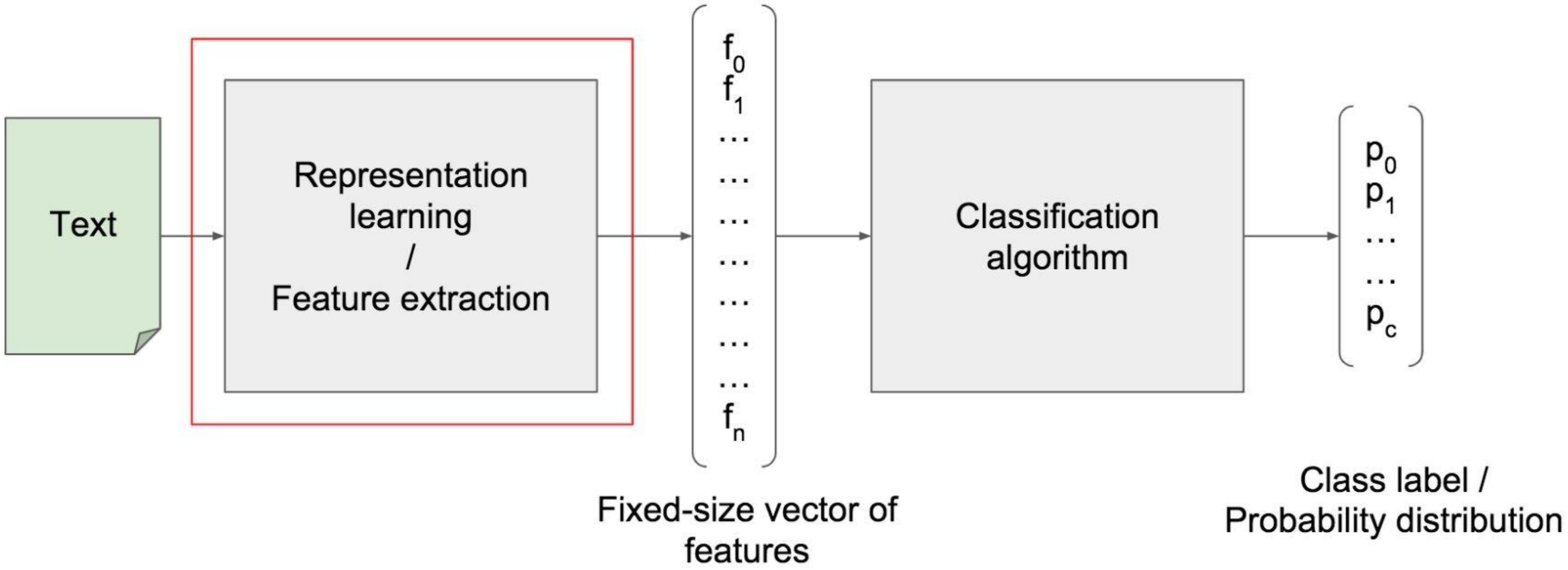
- Discrete labels:
  - Binary: spam filtering, sentiment analysis
  - Multi-class: categorization of items by its description
  - Multi-label: #hashtag prediction
- Continuous labels:
  - Predict product price by its description

Text classification



- Linear Classifier
- Logistic Regression
- etc.

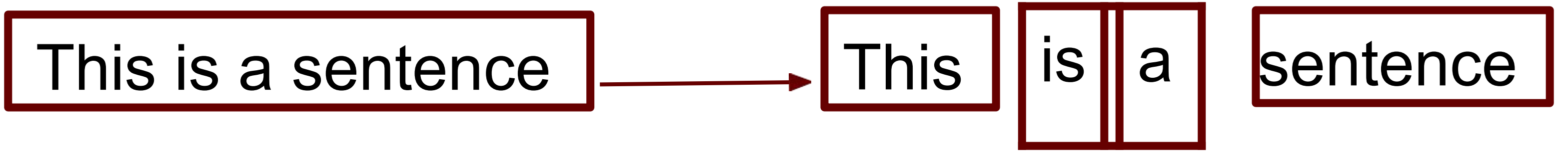
Text classification



Text Preprocessing

Text preprocessing

- Tokenization: split the input into tokens



Text preprocessing

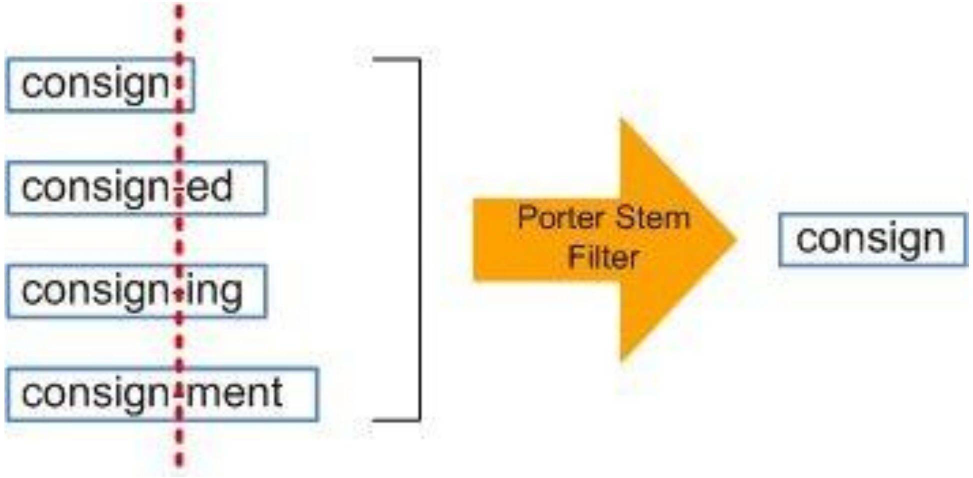
- Token normalization:

Dog, dogs → dog

Bark, barks → bark

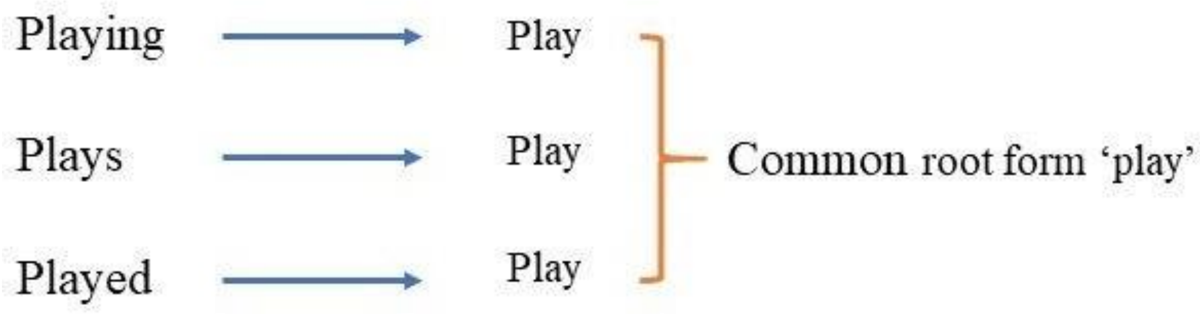
Text preprocessing

- Token normalization:
  - **Stemming**: removing and replacing suffixes to get to the root of the word (**stem**)



Text preprocessing

- Token normalization:
  - **Stemming**: removing and replacing suffixes to get to the root of the word (**stem**)
  - **Lemmatization**: to get base or dictionary form of a word (**lemma**)



Lemmatization

- Lemmatizer:
  - Tries to resolve word to its dictionary form
  - Based on **WordNet** database
  - For the best results feed part-of-speech tagger

Handful tools for preprocessing

- NLTK
  - nltk.stem.SnowballStemmer
  - nltk.stem.PorterStemmer
  - nltk.stem.WordNetLemmatizer
  - nltk.corpus.stopwords
- BeautifulSoup (for parsing HTML)
- Regular Expressions (import re)

Also need to worry about

- Capital Letters
- Punctuation
- Contractions (e.g, etc.)
- Numbers (dates, ids, page numbers)
- Stop-words (“the”, “is”, etc.)
- Tags

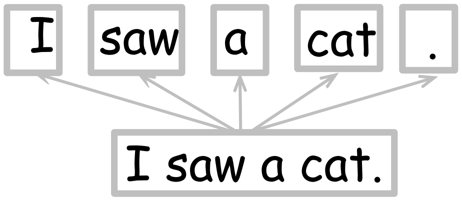
Feature extraction

Why do we need word representations?

I saw a cat.

Text (your input)

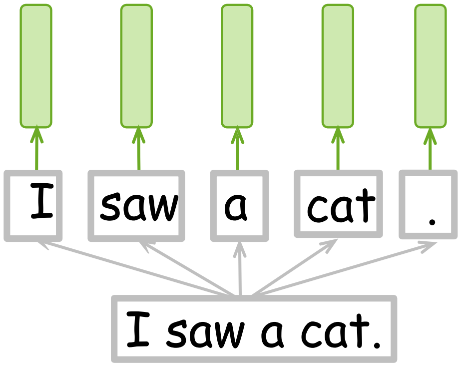
Why do we need word representations?



Sequence of tokens

Text (your input)

Why do we need word representations?

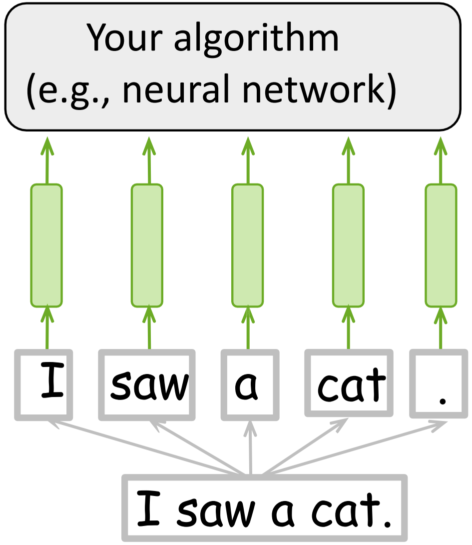


Word representation - vector (input for your model/algorithm)

Sequence of tokens

Text (your input)

Why do we need word representations?



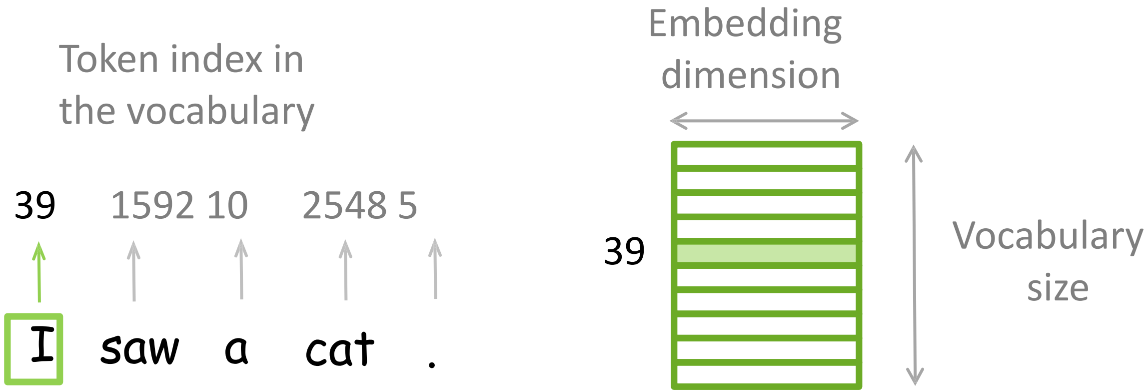
Any algorithm for solving a task

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

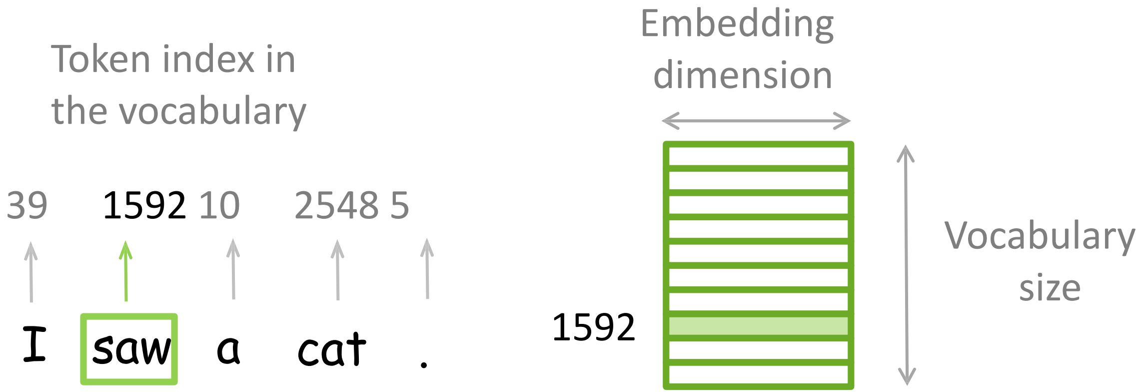
Sequence of tokens

Text (your input)

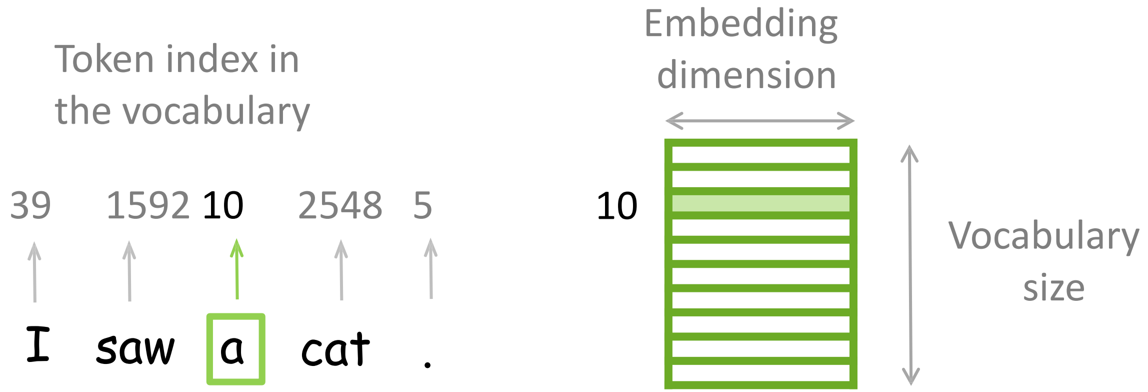
How it works: Look-up Table



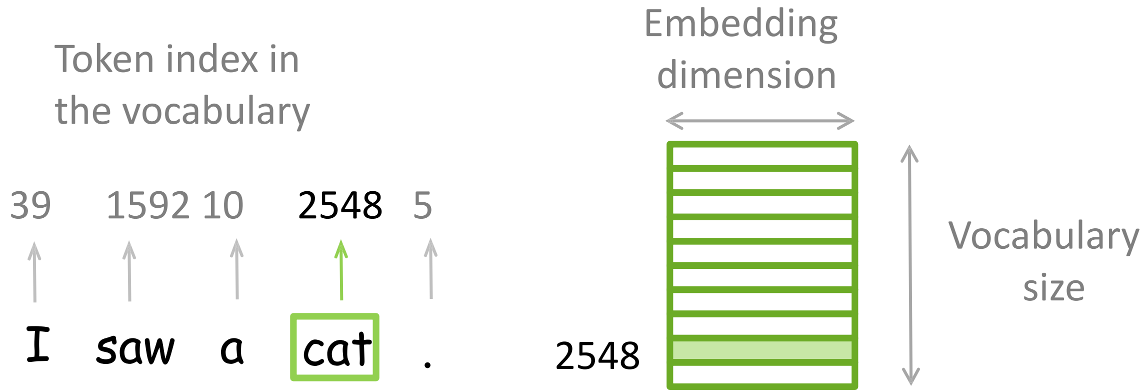
How it works: Look-up Table



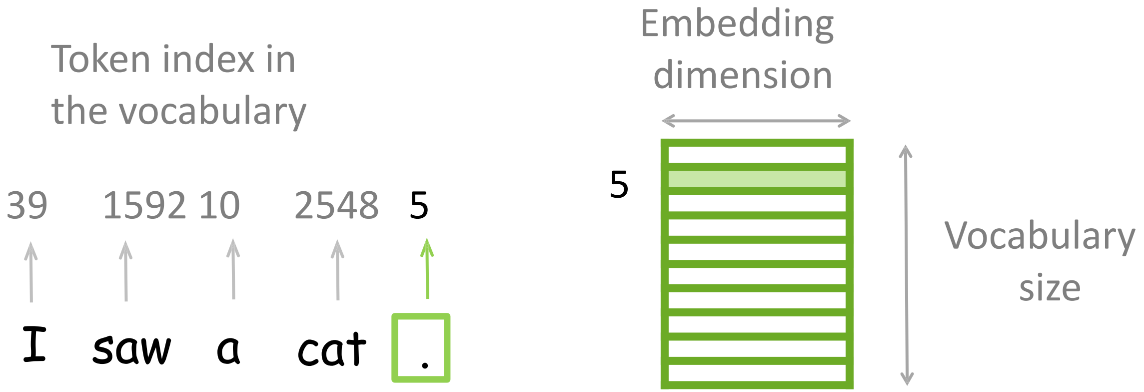
How it works: Look-up Table



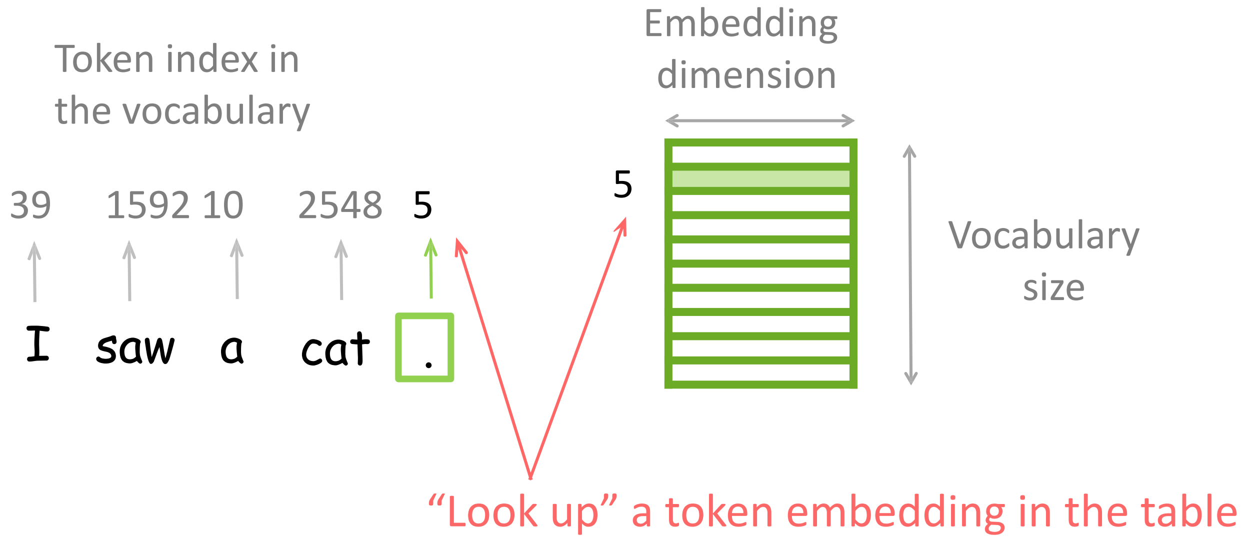
How it works: Look-up Table



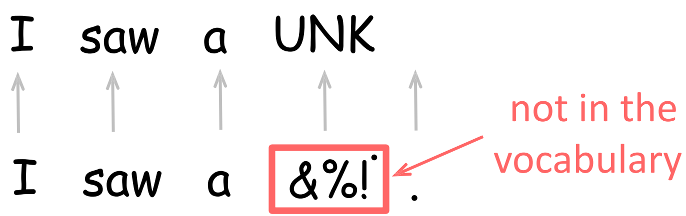
How it works: Look-up Table



How it works: Look-up Table



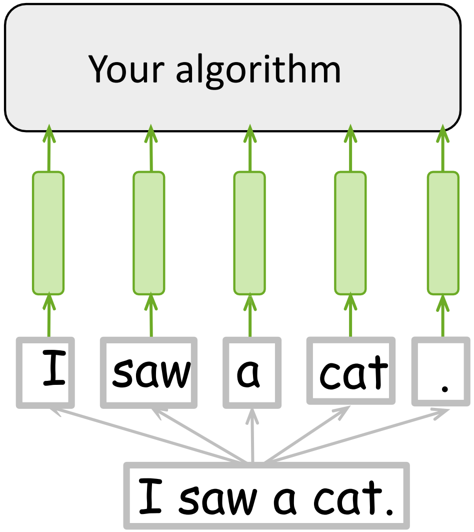
Note UNKs: Out-of-Vocabulary Tokens



Vocabulary is chosen in advance

Therefore, some tokens may be "unknown" – you can use a special token for them

How can we get word representations?



<u>In the following</u>:

How can we get these representations?

One-Hot Vectors: Represent Words as Discrete Symbols

One is 1, the rest are 0

dog 0…0…0 1 0….0…0

cat 0…0 1 0…0….0…0

table 0…0…0….00 1 0…0

Embedding dimension = vocabulary size

One-Hot Vectors: Represent Words as Discrete Symbols

One is 1, the rest are 0

dog	0…0…0 1 0….0…0
cat	0…0 1 0…0….0…0
table	0…0…0….00 1 0…0

Embedding dimension = vocabulary size

Problems:

- Vector size is too large
- Vectors know nothing about meaning

e.g., cat is as close to

dog as it is to table!

One-Hot Vectors: Represent Words as Discrete Symbols

One is 1, the rest are 0

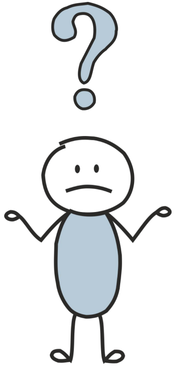
dog	0…0…0 1 0….0…0
cat	0…0 1 0…0….0…0
table	0…0…0….00 1 0…0

Embedding dimension = vocabulary size

Problems:

- Vector size is too large
- Vectors know nothing about meaning

e.g., cat is as close to dog as it is to table!

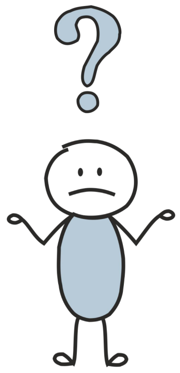


What is meaning?

What is meaning?

Do you know what the word tezgüino means ?

(We hope you do not)



What is meaning?

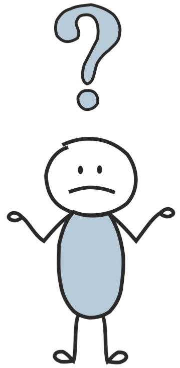
Now look how this word is used in different contexts:

A bottle of tezgüino is on the table. Everyone likes tezgüino.

Tezgüino makes you drunk.

We make tezgüino out of corn.

Can you understand what tezgüino means ?



What is meaning?

Now look how this word is used in different contexts:

A bottle of tezgüino is on the table.

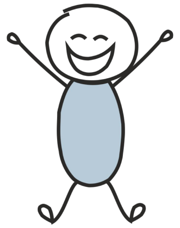
Everyone likes tezgüino.

Tezgüino makes you drunk.

We make tezgüino out of corn.

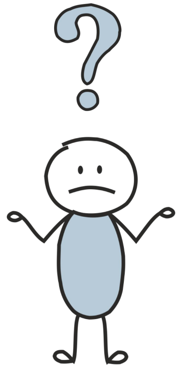
Tezgüino is a kind of alcoholic beverage made from corn.

With context, you can understand the meaning!



What is meaning?

How did you do this?



What is meaning?

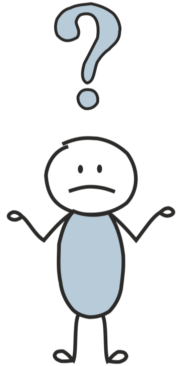
(1) A bottle of ______ is on the table.

(2) Everyone likes ______

(3) ______ makes you drunk.

(4) We make ______ out of corn.

What other words fit into these contexts ?



What is meaning?

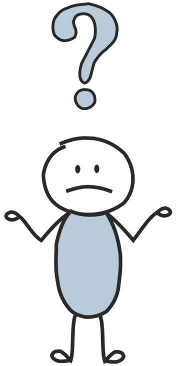
(1) A bottle of ______ is on the table.

(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

What is meaning?

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

(2) Everyone likes ______

(3) ______ makes you drunk.

(4) We make ______ out of corn.

	(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	

This is the distributional hypothesis

rows are similar $\Rightarrow$ meanings of the words are similar

Distributional Hypothesis

Words which frequently appear in similar contexts have similar meaning.

(Harris 1954, Firth 1957)

Main idea:

We have to put information about contexts into word vectors.

Count-Based Methods

Idea: co-occurrence counts

Corpus sentences

He also found five fish swimming in murky water in an old **bathtub**.
We do abhor dust and dirt, and stains on the **bathtub**, and any kind of filth.
Above At the far end of the garden room a **bathtub** has been planted with herbs for the winter.
They had been drinking Cisco, a fruity, wine-based fluid that smells and tastes like a mixture of cough syrup and **bathtub** gin.
Science finds that a surface tension on the water can draw the boats together, like toy boats in a **bathtub**.
In fact, the godfather of gloom comes up with a plot that takes in Windsor Davies (the ghost of sitcoms past), a **bathtub** and a big box of concentrated jelly.
'I'll tell him,' said the Dean from the bathroom above the sound of bathwater falling from a great height into the ample Edwardian **bathtub**.

Co-occurrence counts

the	12
a	9
of	7
and	6
in	5
...	...
like	2
water	2
boat	2
from	2
stain	1
toy	1
god-father	1
Cisco	1
...	...

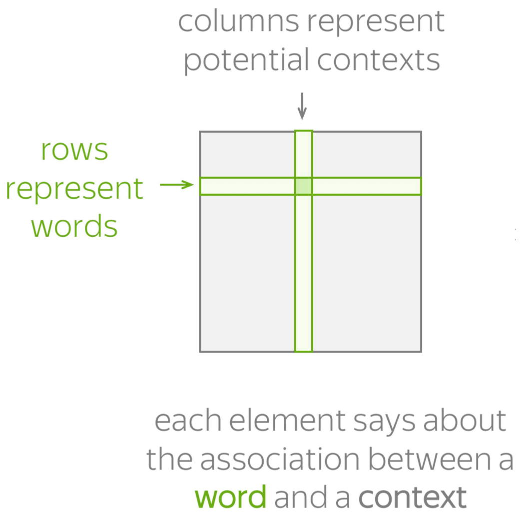
vector

$$\begin{pmatrix} 12 \\ 9 \\ 7 \\ 6 \\ 5 \\ \vdots \\ 2 \\ 2 \\ 2 \\ 2 \\ 1 \\ 1 \\ 1 \\ 1 \\ \vdots \end{pmatrix}$$

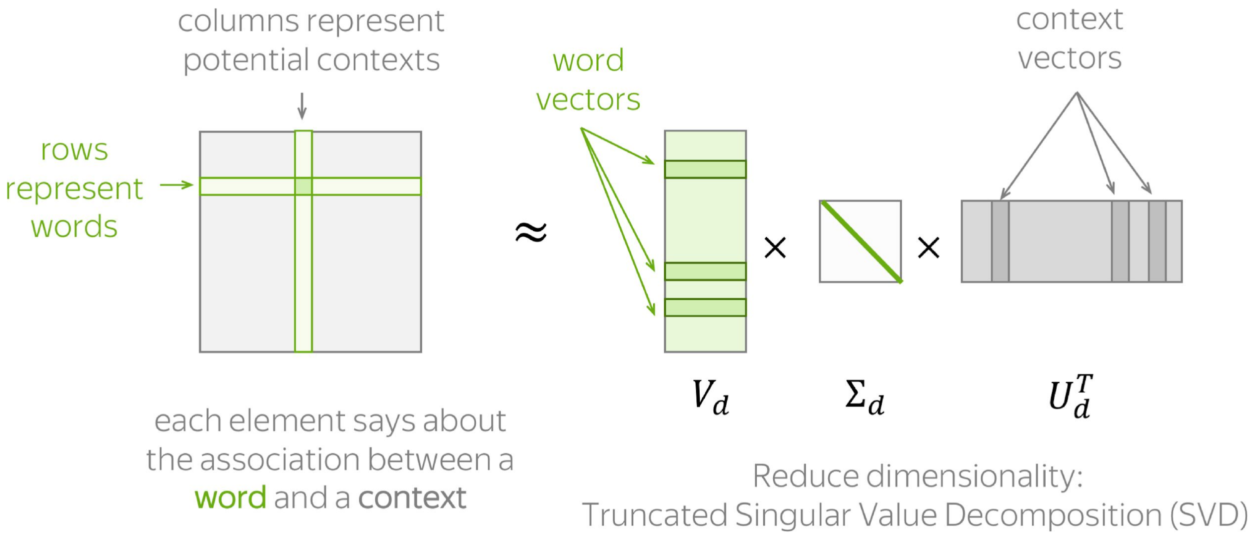
Dimensionality reduction

small vector

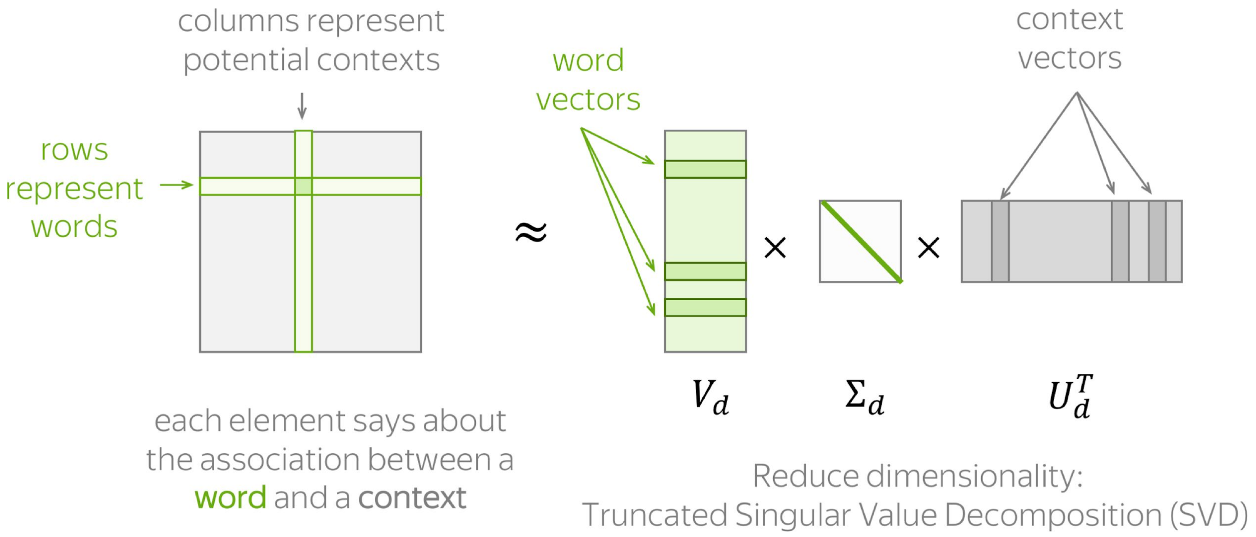
Count-Based Methods: The General Pipeline



Count-Based Methods: The General Pipeline



Count-Based Methods: The General Pipeline



<u>Need to define:</u>

- what is context
- how to compute matrix elements

Simple: Co-Occurrence Counts

2-sized window for cat

… I saw a cute grey cat playing in the garden …

contexts for cat

<u>Context:</u>

- surrounding words
  in a L-sized window

<u>Matrix element:</u>

- number of times word w
  appears in context c

Singular Value Decomposition (SVD)

SVD: $M = U\Sigma V^T \qquad U^T U = I$

- M (m x n) - real matrix
- U (m x m) - orthogonal matrix
- $\sum$ (m x n)- diagonal matrix with non-negative real numbers on the diagonal (singular values)
- V (m x m) - orthogonal matrix

Latent Semantic Analysis (LSA)

Let X be the matrix where t_i is i-th term (word) and d_j is j-th document.

X (vocab_size x corpus_size)

$$
\begin{array}{cc}
 & \begin{array}{ccccc} \phantom{x_{1,1}} & & \mathbf{d}_j & & \phantom{x_{1,n}} \\ & & \downarrow & & \end{array} \\
\begin{array}{c} \\ \\ \mathbf{t}_i^T \rightarrow \\ \\ \\ \end{array} & \begin{bmatrix} x_{1,1} & \dots & x_{1,j} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,1} & \dots & x_{i,j} & \dots & x_{i,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,j} & \dots & x_{m,n} \end{bmatrix}
\end{array}
$$

Latent Semantic Analysis (LSA)

Dot product $t_i^T t_p$ gives correlation between two terms in the corpus, analogically with documents.

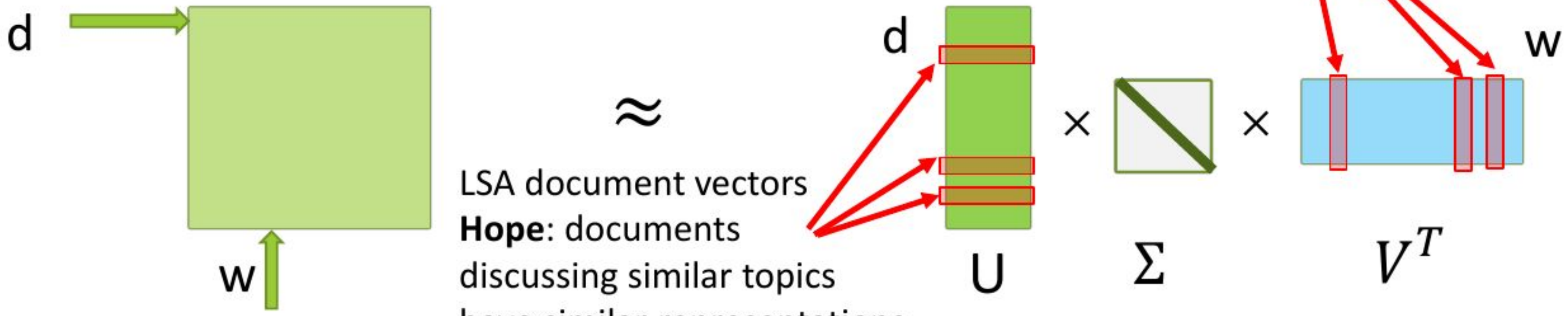
- Let's use SVD $$X = U \Sigma V^T$$
- If you take k biggest singular vectors you will get
  the rank k approximation to X with the smallest error (Frobenius norm)
- Now we can treat terms and documents as semantic space
- Finally we have small representations for terms and documents with meaning
  We can compute similarity with cosine metric.

Latent semantic analysis (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



LSA document vectors
Hope: documents discussing similar topics have similar representations

TF-IDF

Term frequency–inverse document frequency.

We want to represent sentence or document with a vector, based on words we see in this document (also n-grams)

TF-IDF

Term frequency: $$tf(t,d) = \frac{f_{t,d}}{\sum_l f_{l,d}}$$

- $f_{t,d}$ is number of times term t occurs in document d

Another way: $$\mathrm{tf}(t,d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}}$$

- Such view is needed to prevent a bias towards long documents

TF-IDF

Inverse document frequency: $$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

- N - total number of documents in corpus
- Denominator is the number of documents where t appears

TF-IDF

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

$$\mathrm{tf}(t, d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}}$$

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

Probabilities

$$tf(t,d) = \frac{f_{t,d}}{\sum_l f_{l,d}}$$

$$tf(t,d) \sim P(t)$$

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

$$idf(t,D) \sim \log \frac{1}{P(t|d)}$$

$$tfidf(t,d) \sim -P(t)\log(P(t|d))$$

Bag-of-words

Bag-of-words is a model where a text is represented as as bag (multiset) of its words.

We do not consider word order here.

```
Sentence: "John","likes","to","watch","movies","Mary","likes","movies","too"
BoW = {"John":1,"likes":2,"to":1,"watch":1,"movies":2,"Mary":1,"too":1}
```

N-gram Bag-of-words

Instead of words we do the same with n-grams:

```
[
    "John likes",
    "likes to",
    "to watch",
    "watch movies",
    "Mary likes",
    "likes movies",
    "movies too",
]
```