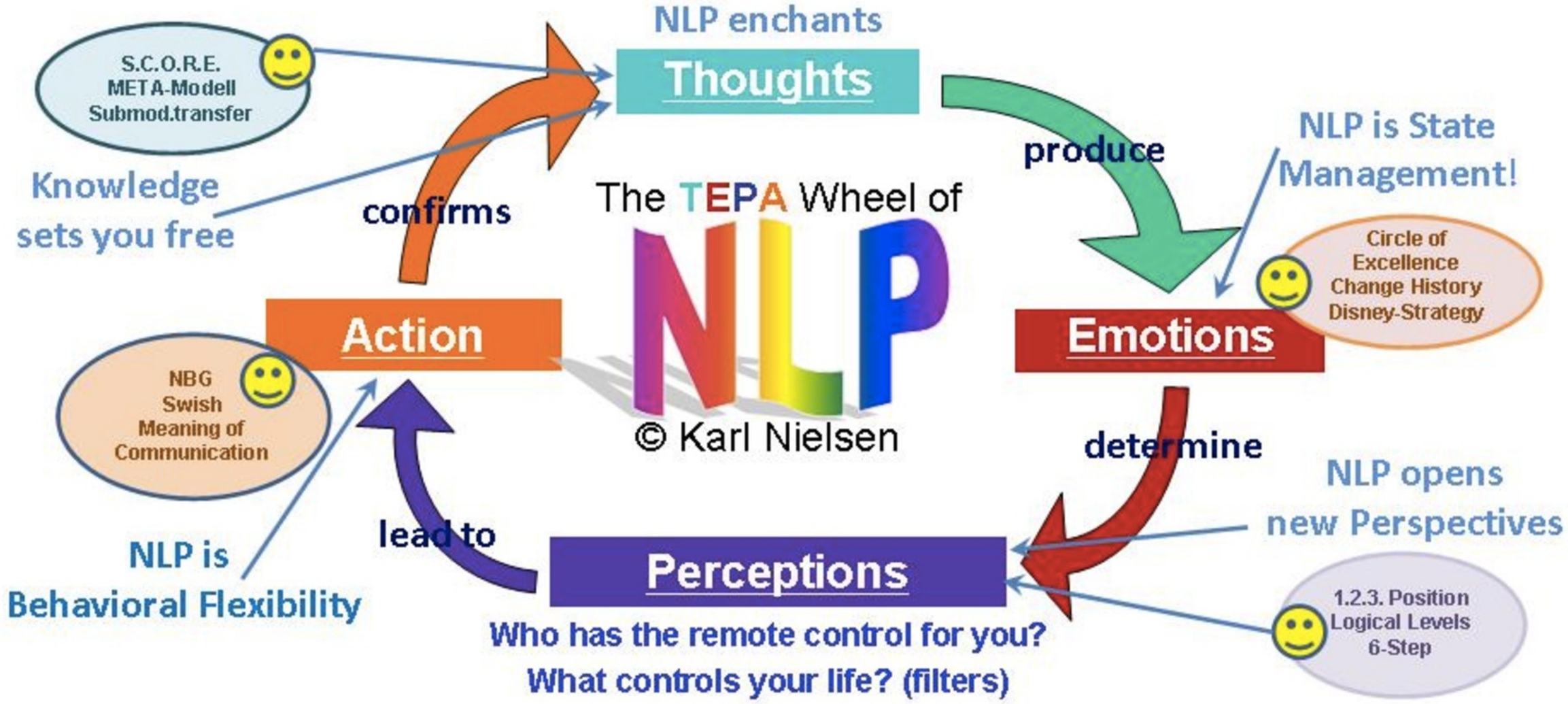
Text Mining and NLP

Ilya Ezepov

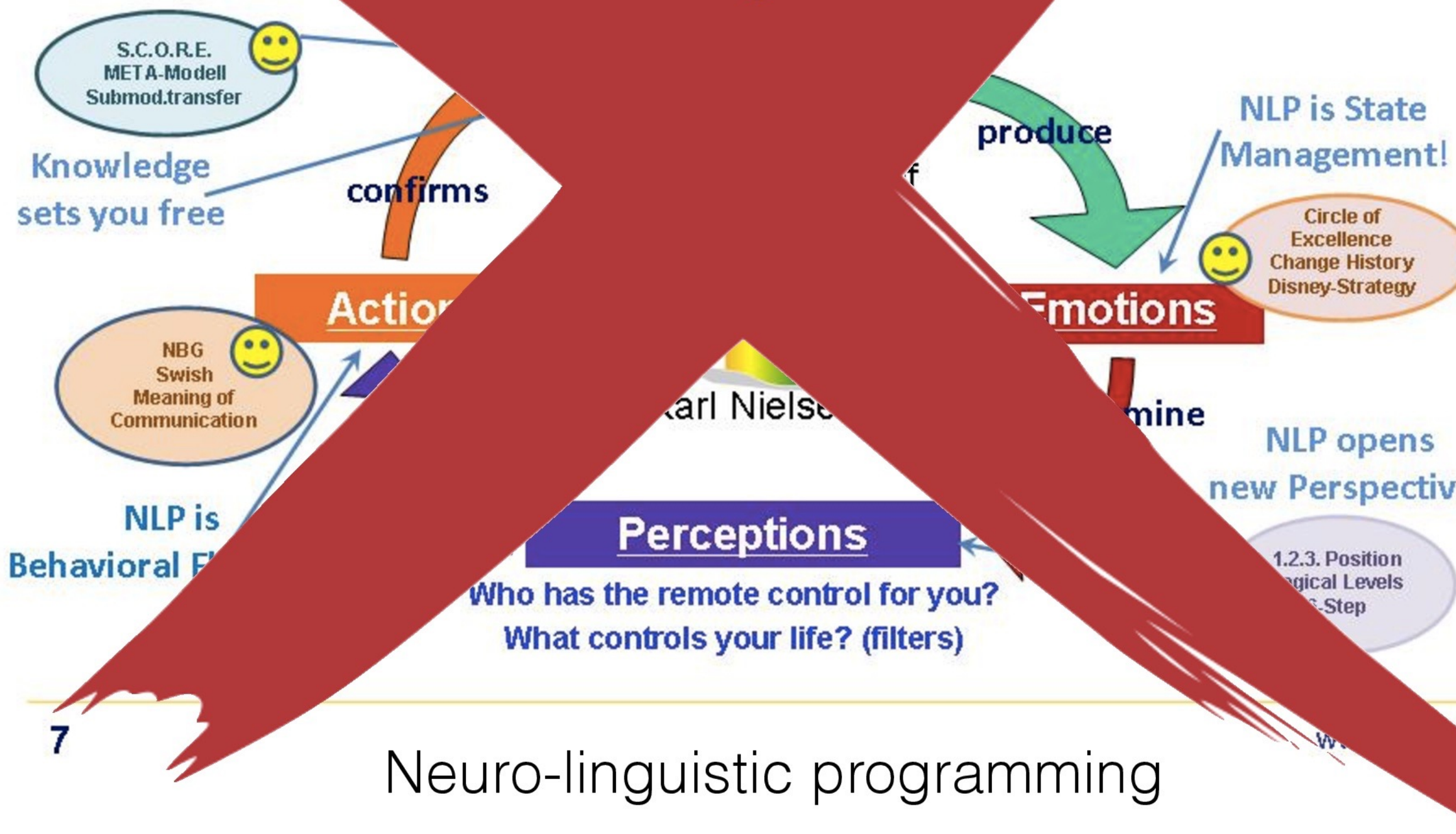
Karl Nielsen

NLP, the freedom in Thinking, Feeling, Perceiving and Behavior



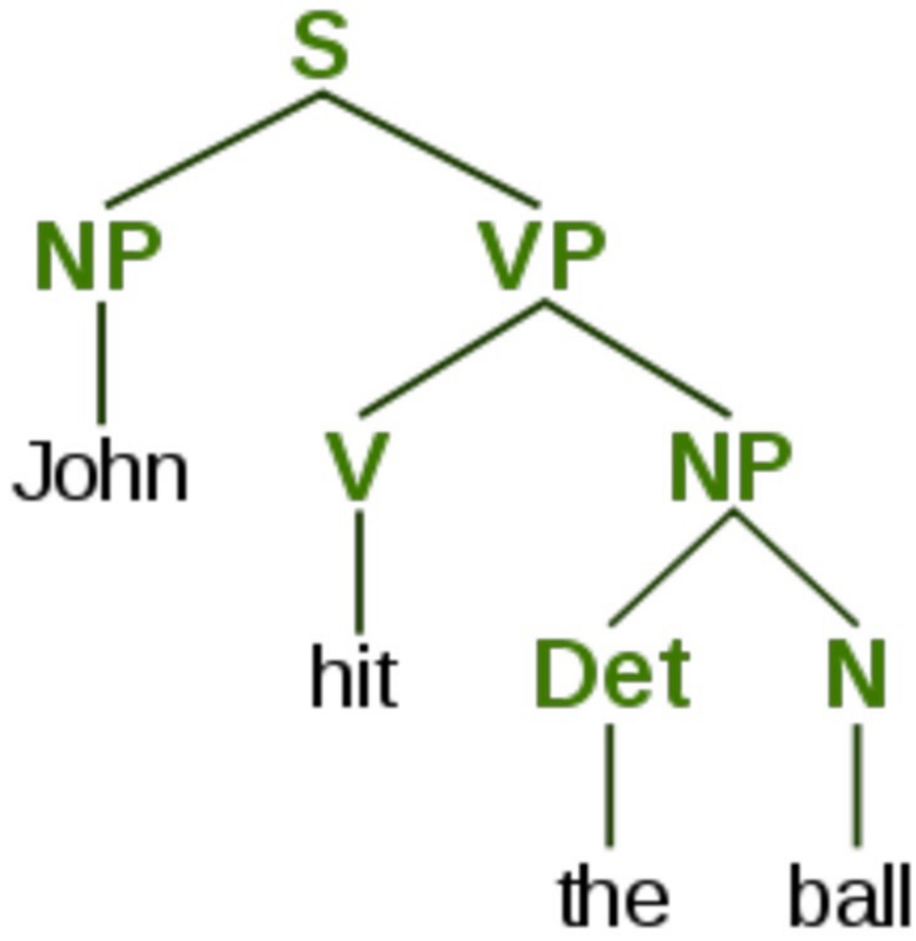
Karl Nielsen

freedom in ...king, Feeli... ...ceiving ...ehavior



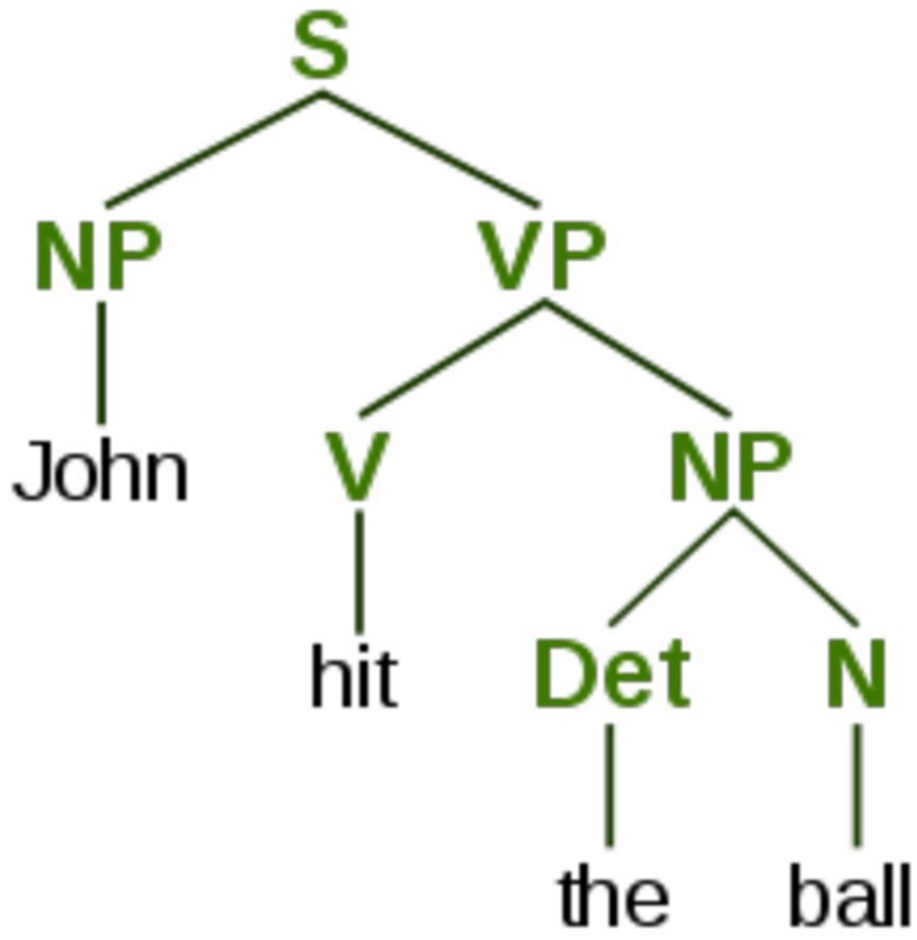
Neuro-linguistic programming

Neuro-linguistic programming



Natural language processing

NLP курильщика



NLP здорового человека

Agenda is boring

Let’s practice!

Kaggle competition:
Bag of Words Meets Bags of Popcorn

- 50,000 IMDB movie reviews
- Binary sentiment: 1 if rating > 6, 0 if rating < 5
- My result: 0.964 AUC, top published: 0.976
- +50,000 unlabelled reviews
- https://www.kaggle.com/c/word2vec-nlp-tutorial/data

How to kaggle it?

Our Data

- Label: 1, Review: *With all this stuff going down at the moment with MJ i've started listening to his music, watching the odd documentary here and there, watched The Wiz and watched Moonwalker again …*

- Label: 0, Review: *The film starts with a manager (Nicholas Bell) giving welcome investors (Robert Carradine) to Primal Park . A secret project mutating a primal animal using fossilized DNA, like ¨Jurassik Park¨, and some scientists resurrect one of nature's most fearsome predators, the Sabretooth tiger or Smilodon …*

- Label: 1, Review: *"The Classic War of the Worlds\" by Timothy Hines is a very entertaining film that obviously goes to great effort and lengths to faithfully recreate H. G. Wells' classic book. Mr. Hines …*

The main idea:

We want to transform non-numerical text data into some numerical form

The main idea:

We want to transform non-numerical text data into some numerical form

== Extract features

The main idea:

We want to transform non-numerical text data into some numerical form

== Extract features

And run some ML on this

Some terminology

- Every single text (review) called ***document***

- Collection of documents is a ***corpus***

Ultimate assumption

- Document is a set of words
- Order of words is not important
- a.k.a. “Bag of words”

Does the word order make a difference?

Does the word order make a difference?

The word order make a difference, does?
Hmmm…

Does the word order make a difference?

The word order make a difference, does?
Hmmm…

Difference make order the a does word?

Does the word order make a difference?

The word order make a difference, does?
Hmmm…

Difference make order the a does word?

What can you say about
Russian vs English?

Text representation

Review: *With all this stuff going down at the moment with MJ i've started listening to his music, watching the odd documentary here and there, watched The Wiz and watched Moonwalker again …*

Lower case
Remove punctuation
Remove html tags

the	3
watched	2
and	2
with	2
wiz	1
listening	1
all	1
at	1
documentary	1
down	1
going	1
here	1
his	1
i've	1
moment	1
mj	1
moonwalker	1
music,	1
odd	1
started	1
...	

Term-document matrix

	the	**watched**	**and**	**a**	**lecture**	**…**
Review 1	4	2	5	3	0	
Review 2	6	0	8	4	0	
…						
Review 50,000	15	1	13	9	1	

… and it's how you convert text into numbers. We're done.

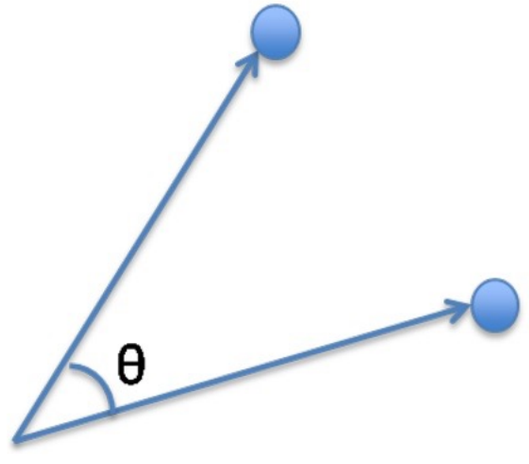
What we able to do now?

- Calculate any statistics about our corpus
- Run any machine learning algorithm
- Calculate documents similarity
- Find document duplicates

Few words about similarity

- Euclid distance is not the best choice
- Length of document vectors has no meaning
- Calculating real distance is high computational cost
- Locality Sensitive Hash: similar objects has similar hashes e.g. element wise *sign* of projection on random plane.

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$



“We’ve had a problem”

25,000 reviews => 75,902 words

“We’ve had a problem”

25,000 reviews => 75,902 words

Term-document matrix is sparse with 1,897,550,000 elements

~3.53Gb with int16 as dense matrix

Sparse, dense?

- Sparse matrix: a lot (95%+) of zeros.
  Can be stored much efficiently (remember only non-zero values and positions).

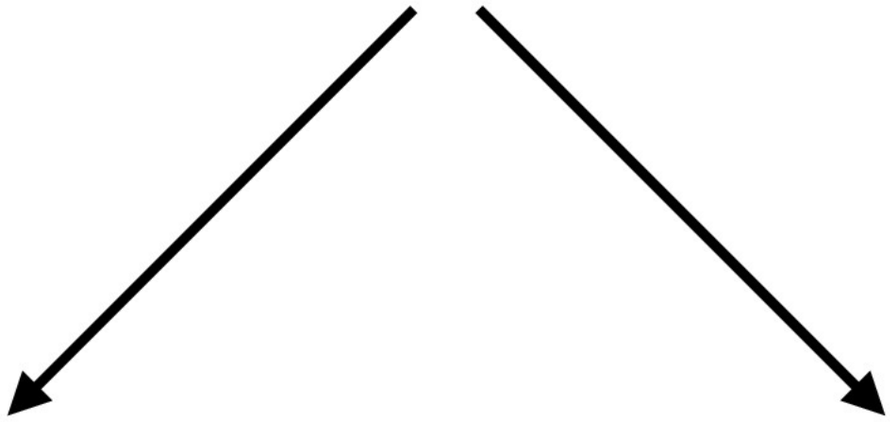
- Dense: few zero values.

- Faster calculations. E.g.: solving system of linear equations is $O(n^2)$ for dense (in parallel), and only O(log n) (typically) for sparse

Why vocabulary is so big?

- Different forms: "argue", "argued", "argues", “arguing" and “argus" ("argument" and “arguments")
- Mistakes: “science”, “sciense”
- Meaningless words (a.k.a. stop-words): “a”, “at”
- Rare words: “physostigmine”

Normal forms



Stemming

Cut off endings and prefixes

Lemmatization

Use vocabulary to get normal forms

Stemming

de facto standard is **Snowball stemmer** (Porter stemmer)

A lot of human-written rules:

- Cut “**’s**”, “**‘**”
- Replace “**sses**” => “**ss**”
- Cut “**ed**”
- Cut “**ing**”
- …

knee	knee
kneel	kneel
kneeled	kneel
kneeling	kneel
kneels	kneel
knees	knee

Pros: no data needed, fast
Cons: a lot of linguist pre-work, mistakes (“better”/“good”)

Lemmatization

Lookup words in the dictionary

Pros: Tolerance to “better”/“good”, can resolve with context
Cons: external data is needed, relatively slow

Best for Russian language:
python-library **pymorphy** (100K words/sec)

pymorphy: https://habrahabr.ru/post/49421/
pymorphy2: https://habrahabr.ru/post/176575/

Apply this to our problem

No stemming:
25,000 reviews => 75,902 words

Stemming:
25,000 reviews => **72,578** words

Not really great success

Why vocabulary is so big?

- ~~Different forms: "argue", "argued", "argues", “arguing" and “argus" ("argument" and “arguments")~~
- Mistakes: “science”, “sciense”
- Meaningless words (a.k.a. stop-words): “a”, “at”
- Rare words: “physostigmine”

Stop-words

- Words like “and”, “a”, ”the”, “on”, …
- Can be found in (particularly) any text
- Have mo meaning
- We don’t want to build our sentiment model on this
- Let’s drop them from the text!
- Two classical ways:
  - Use a list of stop-words
  - Drop words, that are in more than 90% of documents

How about Russian “не”?

Apply this to our problem

No stemming:
25,000 reviews => 75,902 words

Stemming:
25,000 reviews => 72,578 words

Stemming + Stop-words (90% of docs):
25,000 reviews => **72,573** words

Were dropped: 'and', 'it', 'of', 'the', 'to'

Rare words

- Misspellings
- Neologisms
- Bugs in the data
- Incorrect text preparation/normalization
- Let's drop words that are less than in 0.1% of documents (25 reviews in our case)

Apply this to our problem

Stemming:
25,000 reviews => 72,578 words

Stemming + Stop-words (90% of docs):
25,000 reviews => 72,573 words

Stemming + Stop-words + Rare words:
25,000 reviews => **9,232** words

Our term-document matrix is now only 83Mb

Metaphones/Soundex

- Cool algorithm for fixing mistakes
- Let's convert our words into transcriptions (how word sounds)
- Most of mistakes are made because people write a word how it sounds. We can fix these mistakes.

Щавель → щ'ав'эл'

Щявель → щ'ав'эл'

Шявель → щ'ав'эл'

Щавиль → щ'ав'эл'

Levenshtein distance

- Just for general knowledge
- Measure of distance between two words (strings in general)
- Minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other
- Lev(cat, cats) = 1, Lev(cat, dog) = 3
- We can somehow use it to detect mistakes

Prove that it’s a distance!

Why vocabulary is so big?

- ~~Different forms: "argue", "argued", "argues", “arguing" and “argus" ("argument" and “arguments")~~
- ~~Mistakes: “science”, “sciense”~~
- ~~Meaningless words (a.k.a. stop-words): “a”, “at”~~
- ~~Rare words: “physostigmine”~~

But solution is not cool!

tf-idf

Very important idea about
normalization of term-document matrix

tf: term frequency,
normalize matrix by rows

idf: inverse document frequency,
normalize matrix by columns

tf

- Term frequency: how to count words in the document?
    - Counts (we already at this)
    - Frequency (canonical tf)
    - Boolean frequency (1/0)
    - log(Frequency)

tf example

	the	watched	and	a	lecture
Review 1	4	2	0	3	0
Review 2	2	0	8	4	1
Review 3	1	2	0	0	1

Applying tf…

	the	watched	and	a	lecture
Review 1	0.44	0.22	0	0.33	0
Review 2	0.13	0	0.53	0.27	0.07
Review 3	0.25	0.50	0	0	0.25

idf

- Inverse document frequency: increase weight of rare words
    - idf = 1 (we already at this)
    - $\text{idf} = \ln\left(\frac{N}{n_t}\right)$ (canonical idf)
    - $\text{idf} = \ln\left(1 + \frac{N}{n_t}\right)$
    - Invent yours!

idf example

	the	watched	and	a	lecture
Review 1	4	2	0	3	0
Review 2	2	0	8	4	1
Review 3	1	2	0	0	1

Applying idf…

	the	watched	and	a	lecture
Review 1	0	0.81	0	1.22	0
Review 2	0	0	8.89	1.62	0.41
Review 3	0	0.81	0	0	0.41

tf-idf example

	the	watched	and	a	lecture
Review 1	4	2	0	3	0
Review 2	2	0	8	4	1
Review 3	1	2	0	0	1

Applying tf*idf…

	the	watched	and	a	lecture
Review 1	0	0.18	0	0.40	0
Review 2	0	0	4.71	0.53	0.03
Review 3	0	0.41	0	0	0.10

Why tf-idf?

- Document size doesn't matter any more!
- Stop-words are killed by idf!
- Rare words are killed by tf!

Why tf-idf?

- Document size doesn't matter any more!
- Stop-words are killed by idf!
- Rare words are killed by tf!

And we can do it with MapReduce!

(do you remember our first lecture? Let's recap)

Calculate tf with MR

Data:

doc1: word1 word2 word1
doc2: word1
doc3: word1 word2

Map output:

doc_word, 1

1_word1, 1
1_word2, 1
1_word1, 1

2_word1, 1
3_word1, 1
3_word2, 1

Reduce output:

f(x) = len(x)

doc_word, counted

1_word1, 2
1_word2, 1
2_word1, 1
3_word1, 1
3_word2, 1

TF counted!

Calculate idf with MR

Data:

doc1: word1 word2 word1
doc2: word1
doc3: word1 word2

Map output:

word, doc

word1, 1
word2, 1
word1, 1

word1, 2
word1, 2
word1, 3

Reduce output:
f(x) = len(uniq(x))

word, counted

word1, 3
word2, 2

IDF counted!

What about tf-idf?

- We are not done, because both results (tf and idf) can be more than RAM
- Let passthrough!

No problems with tf-idf!

- We are not done, because both results (tf and idf) can be more than RAM
- Let passthrough!

TF:
1_word1, 2
1_word2, 1
2_word1, 1
3_word1, 1
3_word2, 1

Map:
doc_tf
word1, 1_2
word2, 1_1
word1, 2_1
word1, 3_1
word2, 3_1

Reduce:
df_doc_tf
word1, 3_1_2
word2, 2_1_1
word1, 3_2_1
word1, 3_3_1
word2, 2_3_1

How to get better model?

- Now we are using words as text features (***terms***)
- Why not to try pairs of words?
- In general: ***n-gramps*** – *sequences of n words*
- Maybe n-gramms of chars?
(great for language detection)

It really increases results.
But dictionary is going to explode!

How to get better model?

- Now we are using words as text features (***terms***)
- Why not to try pairs of words?
- In general: ***n-gramps*** – *sequences of n words*
- Maybe n-gramms of chars?
  (great for language detection)

Document: "*I thought it was not the best recap episode*"

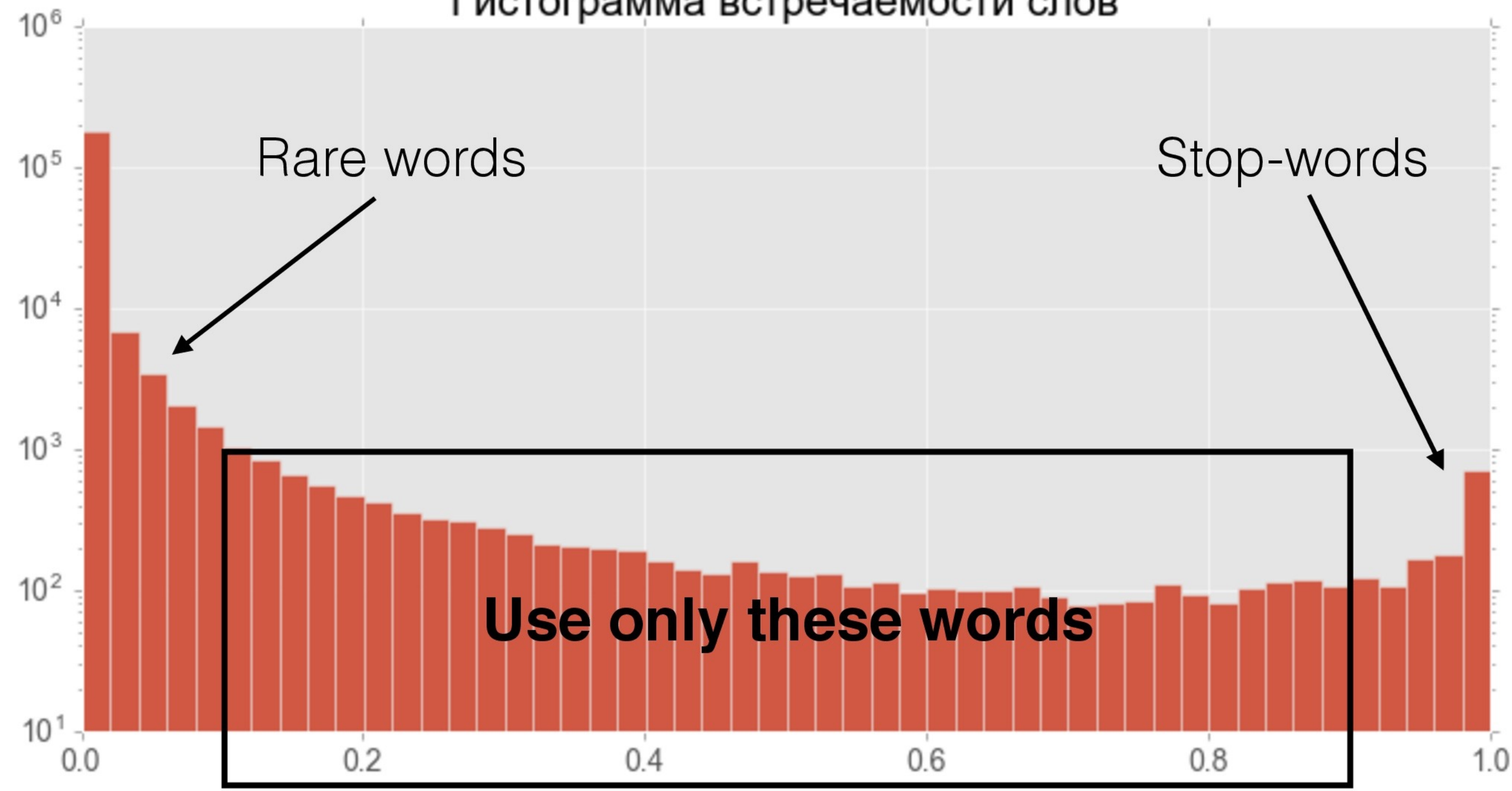
Vocabulary on 1-gramms and 2-gramms:

I, though, it, was, not, the, best, recap, episode,
I thought, thought it, it was, was not, not the, the best, best recap, recap episode

My current thesis

- Data: RBC, Lenta, Interfax, 2009 — 2014
- Lemmatization
- 1-gramm of words
- 10% – 90% thresholds for stop/rare words
- tf-idf
- Logistic regression to predict RTS daily return

My current thesis



My current thesis

Top negative words:

асад, нургалиев, клепач, бородин, олланд

Top positive words:

пиво, болельщик, цска, ураган, матч

Order makes a difference

No bag of words any more

Words embeddings

- We want to **understand** words
- Let's transform (embed) out words to vectors in some abstract multidimensional space (N dims)
- Start with random embedding
- And we want to somehow train these embeddings

Embeddings

- aka Representation learning
- General concept: representing some categorical data with dense vector

Word	Embedding
king	(0.188, 0.325, …, 0.708, 0.656)
man	(-0.471, 0.356, …, -0.806, 1.001)
cmf	(0.117, 0.094, …, 1.178, -0.530)
Lomonosov	(-0.057, 1.587, …, -0.021, -0.077)

word2vec

- Developed by Google
- Problem: We want to predict words, based on context
- Representing words with abstract high-dimensional vector
- Needs lot of data

Skip-gram algorithm

*To Sherlock Holmes **she** always was the woman.*

1. For every word in the text

Skip-gram algorithm

*To Sherlock Holmes **she** always was the woman.*

window ±2

2. Determine window

Skip-gram algorithm

*To Sherlock Holmes **she always** was the woman.*

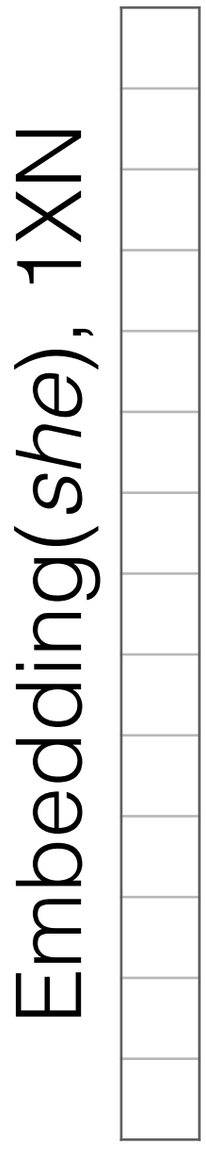
window ±2

3. Select random **target** in the window

Skip-gram algorithm

*To Sherlock Holmes **she** **always** was the woman.*

window ±2

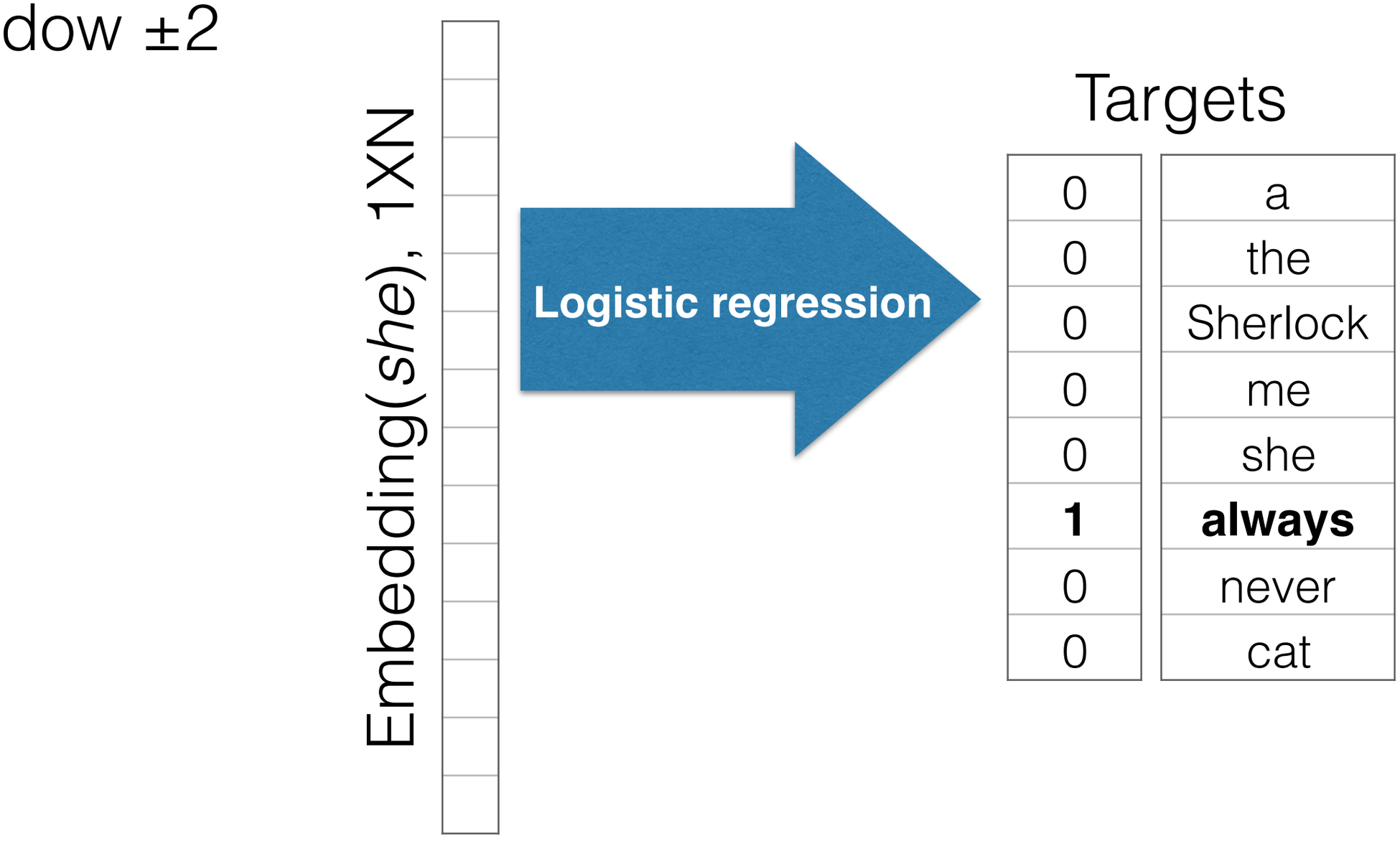


4. Looking up embedding

Skip-gram algorithm

*To Sherlock Holmes **she** **always** was the woman.*

window ±2

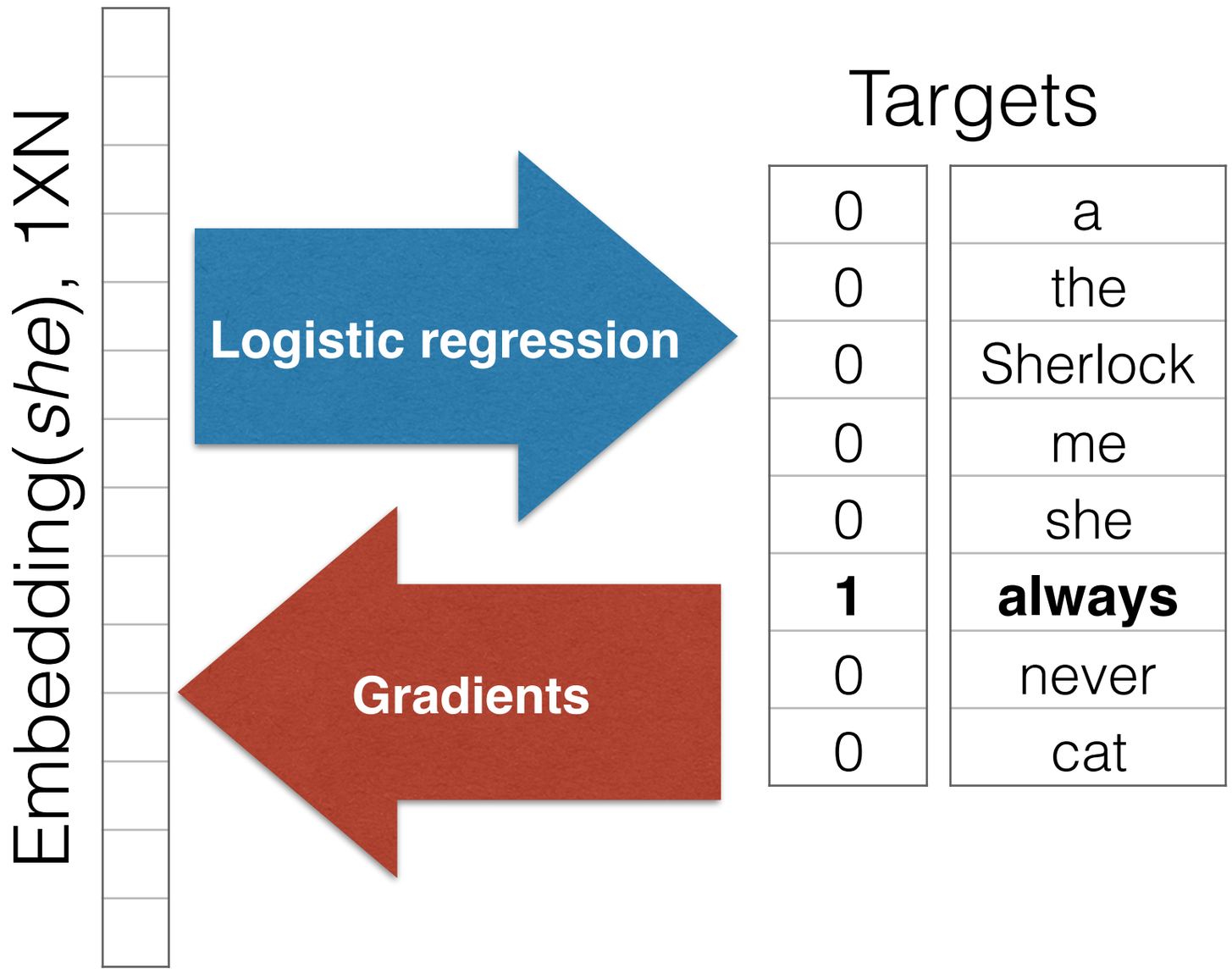


5. Using linear regression to predict target

Skip-gram algorithm

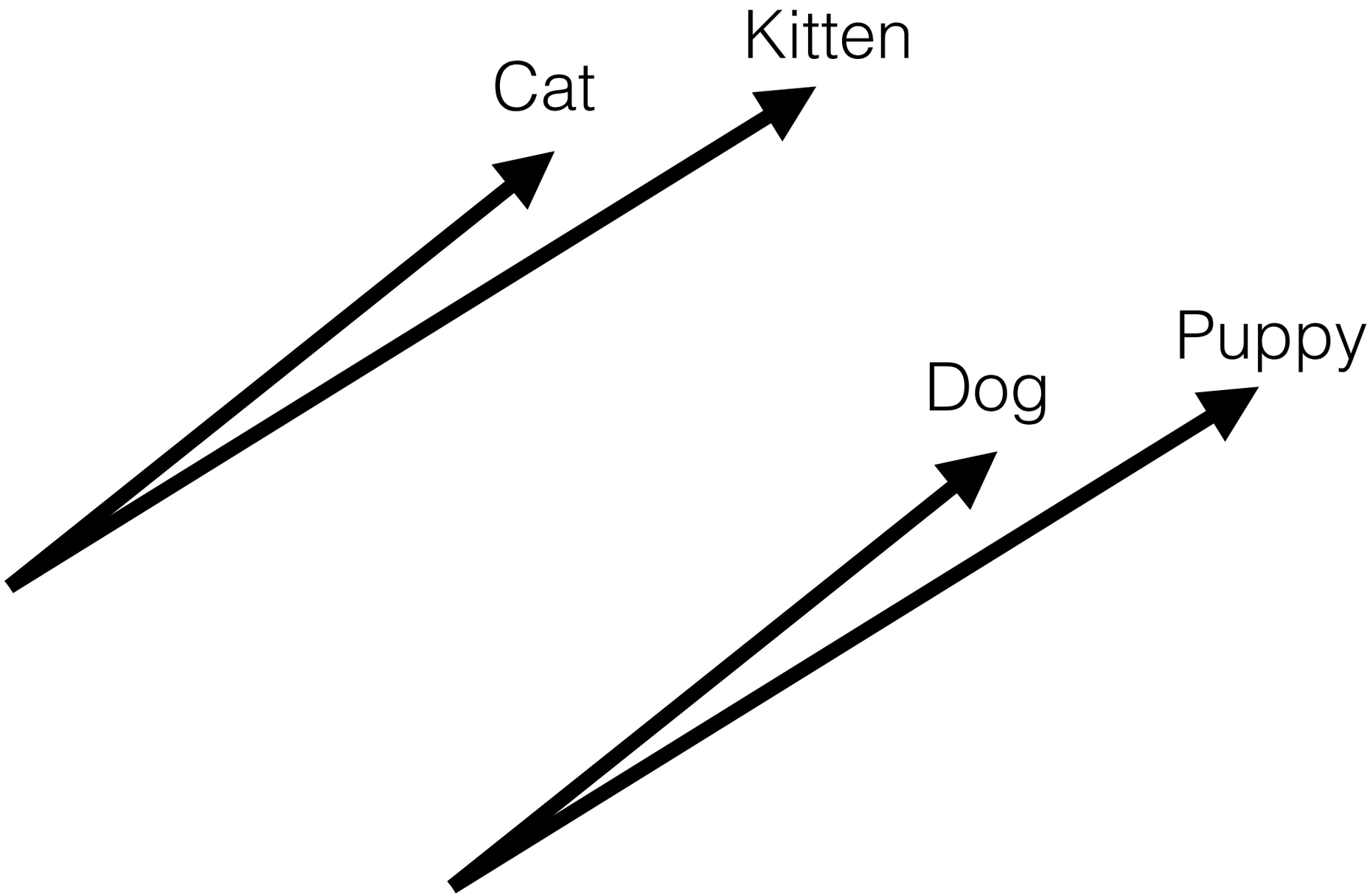
*To Sherlock Holmes **she** **always** was the woman.*

window ±2



6. Updating weights in regression **+ updating embeddings**

Now every word is vector!



Word2vec examples

You can do to words anything you can do to vectors!

- Find closest & cluster: cat, kitty, kity, tiger, lion, …
- Find irrelevant: London, Russia, Paris, Moscow
- Do algebra!

Grammar: Short -> Shortest … Good -> ?

Grammar: Short -> Shortest … Good -> ?

Good - Short + Shortest = V
Find closest word to V

Grammar: Short -> Shortest … Good -> ?

Good - Short + Shortest = V
Find closest word to V

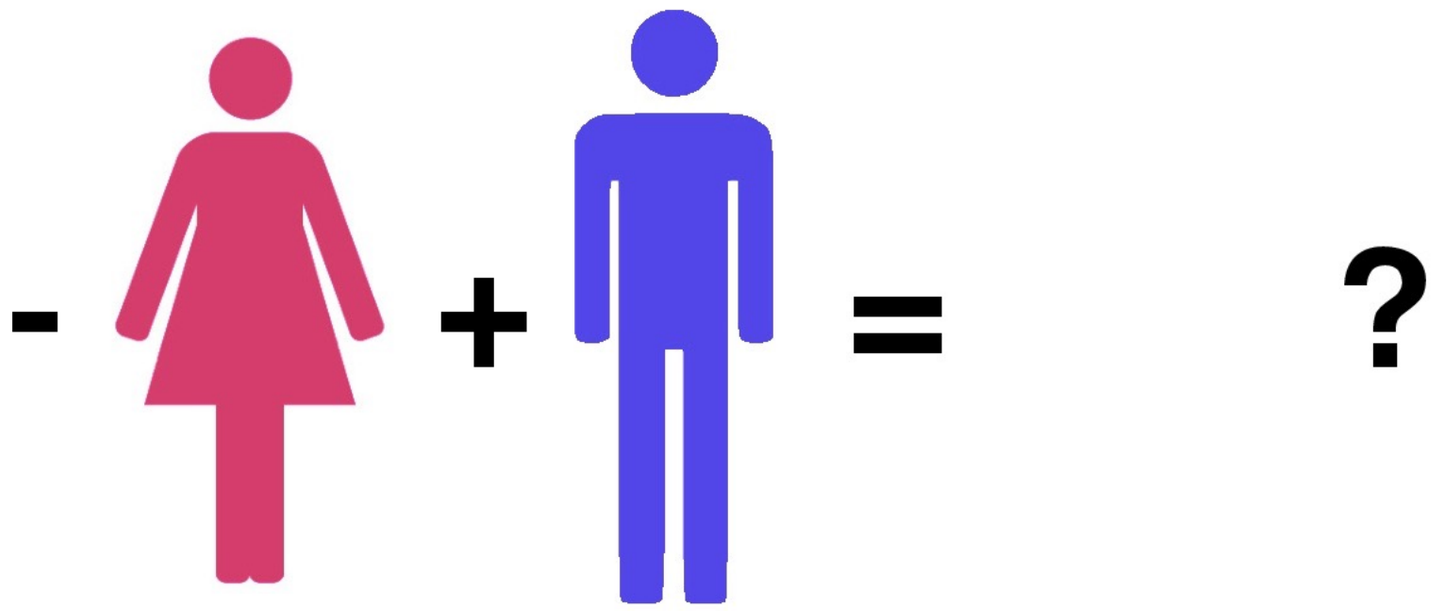
Short -> Shortest … Good -> Best
Be -> Was … Read -> Read

Grammar: Short -> Shortest … Good -> ?

Good - Short + Shortest = V
Find closest word to V

Short -> Shortest … Good -> Best
Be -> Was … Read -> Read

Meaning:

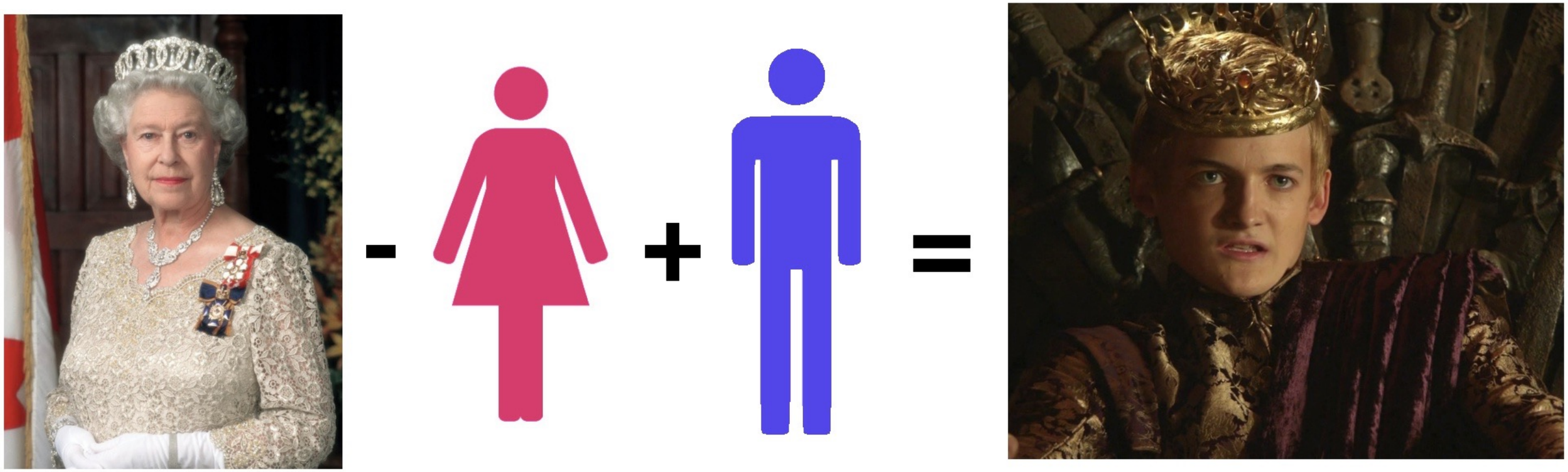


Grammar: Short -> Shortest … Good -> ?

Good - Short + Shortest = V
Find closest word to V

Short -> Shortest … Good -> Best
Be -> Was … Read -> Read

Meaning:

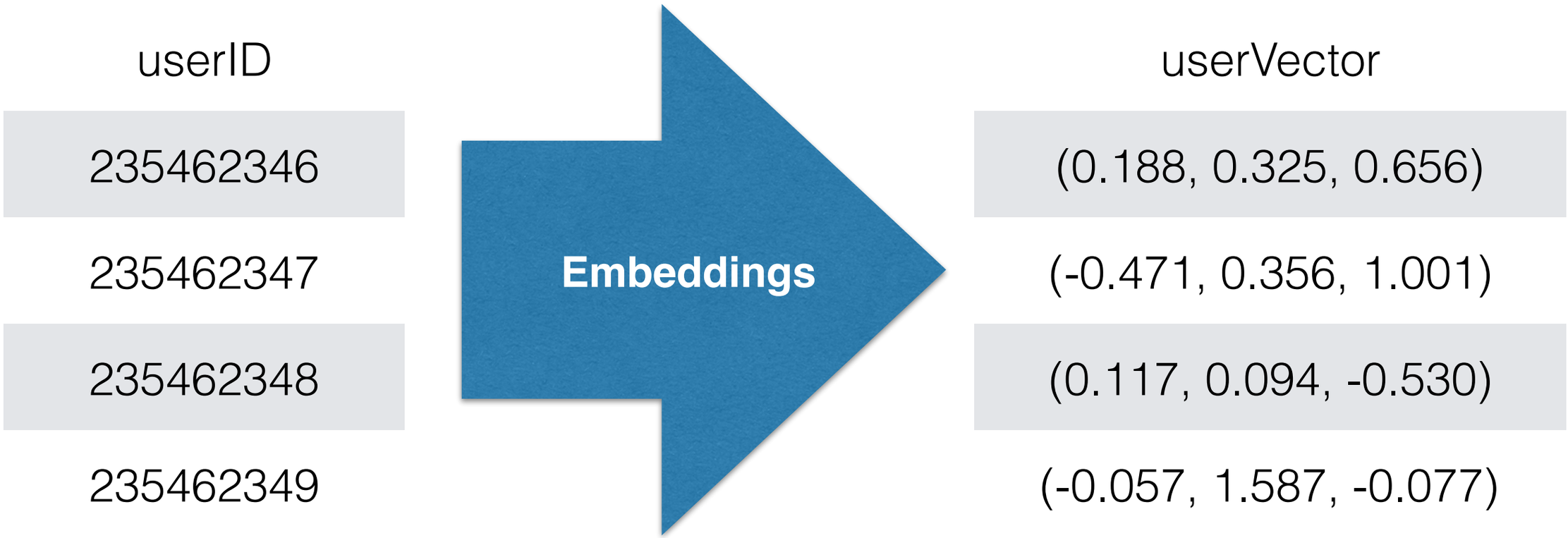


something2vec

- Any sequence can be embedded
- Bank transactions
- Web sessions
- Users graph
- …you name it

Embedding layer

- New layer for Neural Networks!
- Any categorical variable can be embedded



The end