Text Classification

Credit of the slides goes to Lena Voita and Standford CS

DATE : 30/09/2024

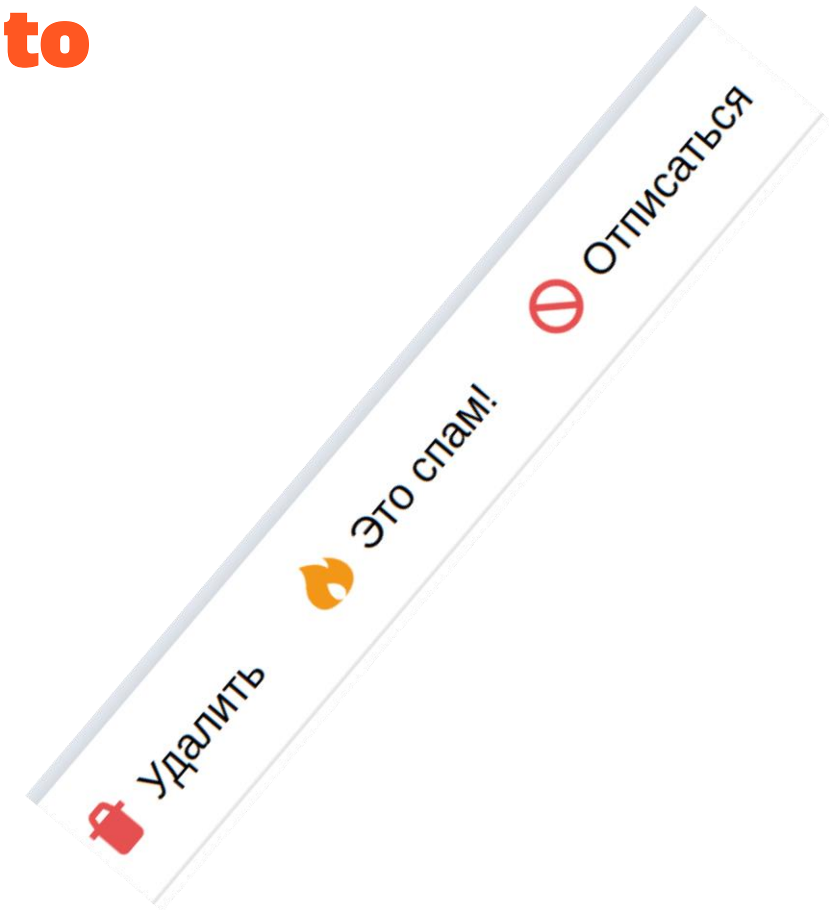
Why do we need to classify texts?

Why do we need to classify texts?

- As a self-sufficient task:

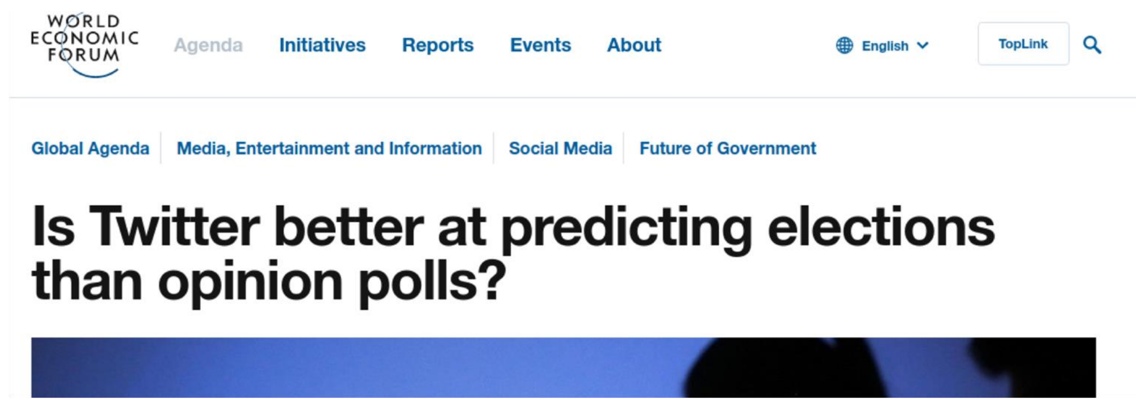
Why do we need to classify texts?

- As a self-sufficient task:
  - Spam filtering



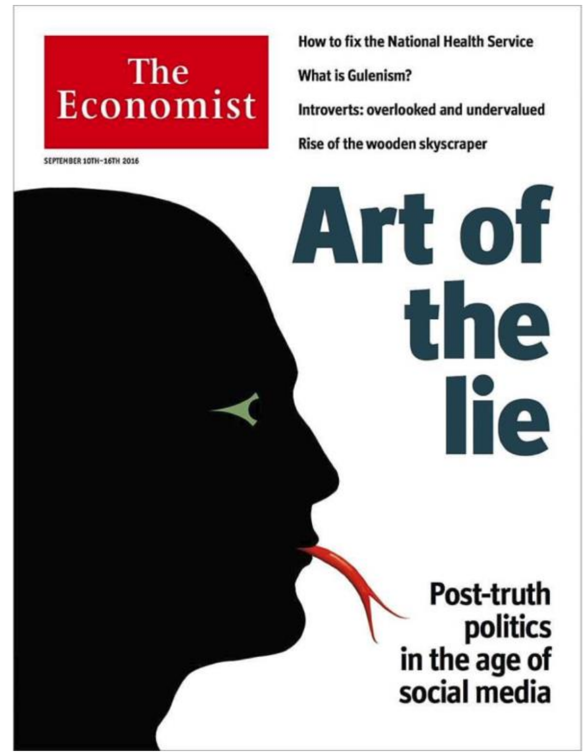
Why do we need to classify texts?

- As a self-sufficient task:
  - Spam filtering
  - Sentiment analysis



Why do we need to classify texts?

- As a self-sufficient task:
  - Spam filtering
  - Sentiment analysis
  - Fake news/clickbait detection
  - Troll/bot protection



Why do we need to classify texts?

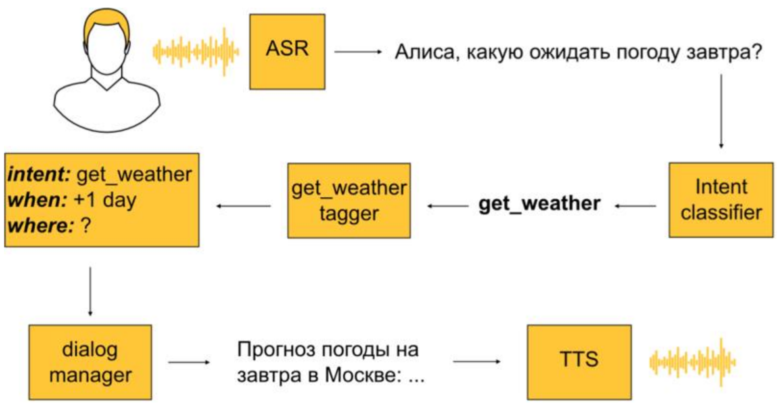
- As a self-sufficient task:
  - Spam filtering
  - Sentiment analysis
  - Fake news/clickbait detection
  - Troll/bot protection
- As a part of more complicated NLP tasks

Why do we need to classify texts?

- As a self-sufficient task:
  - Spam filtering
  - Sentiment analysis
  - Fake news/clickbait detection
  - Troll/bot protection
- As a part of more complicated NLP tasks
  - Data filtering

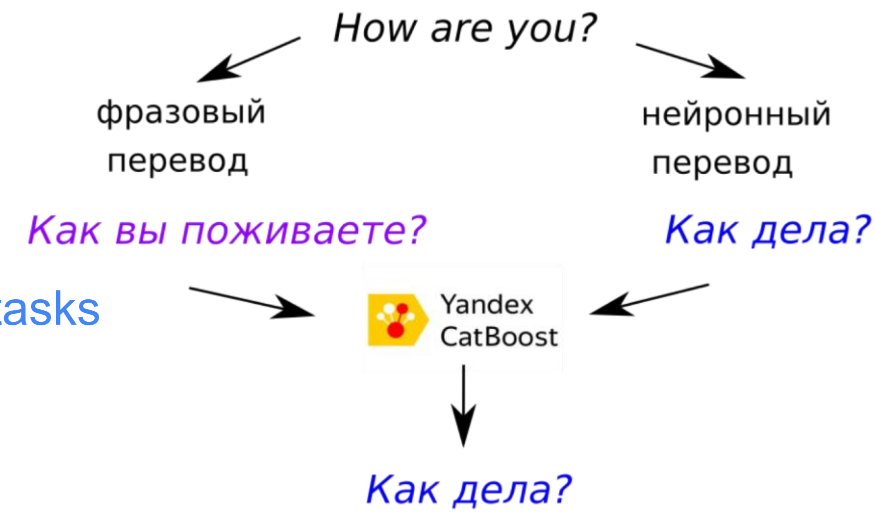
Why do we need to classify texts?

- As a self-sufficient task:
  - Spam filtering
  - Sentiment analysis
  - Fake news/clickbait detection
  - Troll/bot protection
- As a part of more complicated NLP tasks
  - Data filtering
  - Intent classification in dialog systems



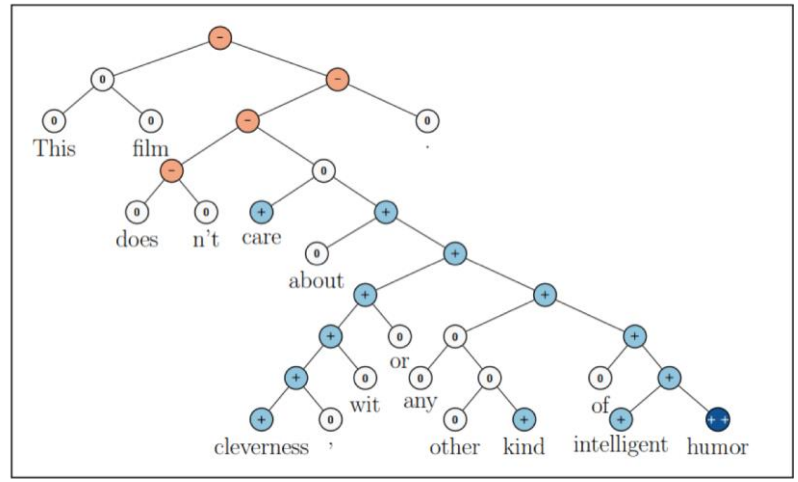
Why do we need to classify texts?

- As a self-sufficient task:
  - Spam filtering
  - Sentiment analysis
  - Fake news/clickbait detection
  - Troll/bot protection
- As a part of more complicated NLP tasks
  - Data filtering
  - Intent classification in dialog systems
  - Hybrid machine translation systems :-)



Popular Benchmarks

Dataset	Size
Question classification (TREC)	6K
MPQA Opinion corpus (SUBJ)	8K
Movie Reviews (MR)	10K
Reuters-21578	21.5K
IMDB Reviews	25K
Stanford Sentiment Treebank (SST)	9.5K
Sogou News	0.5M
AG News	1M
Yelp Dataset	5.2M
Amazon Reviews	35M
Flickr 100m	100M

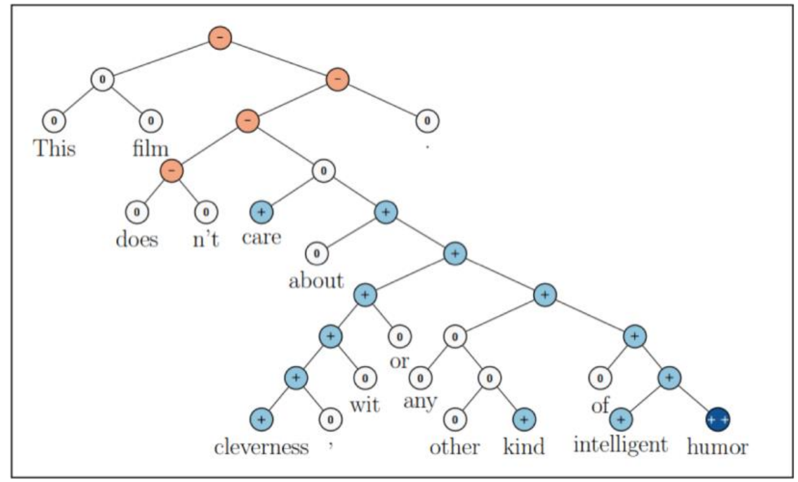


Subjective unigram	Objective unigram
amazing, beautiful, cheap, decent, effective, fantastic, good, happy, impress, jittery, light, madly, nice, outstanding, perfect, quick, responsive, sharp, terrible, ultimate, wonderful.	access, because, chance, default, entire, few, go, half, inside, job, keep, know, last, matter, new, only, past, quality, read, several, text, use, version, was, young.

Popular Benchmarks

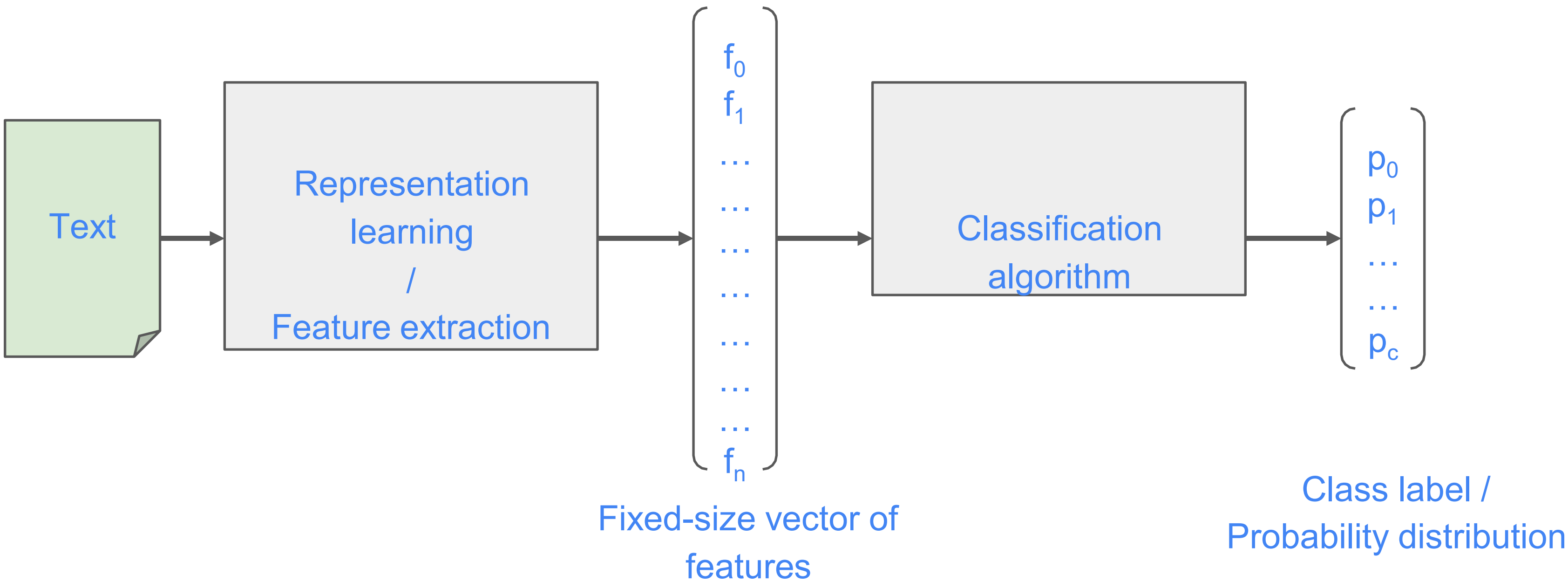
Dataset	Size
Question classification (TREC)	6K
MPQA Opinion corpus (SUBJ)	8K
Movie Reviews (MR)	10K
Reuters-21578	21.5K
IMDB Reviews	25K
Stanford Sentiment Treebank (SST)	9.5K
Sogou News	0.5M
AG News	1M
Yelp Dataset	5.2M
Amazon Reviews	35M
Flickr 100m	100M

~2012

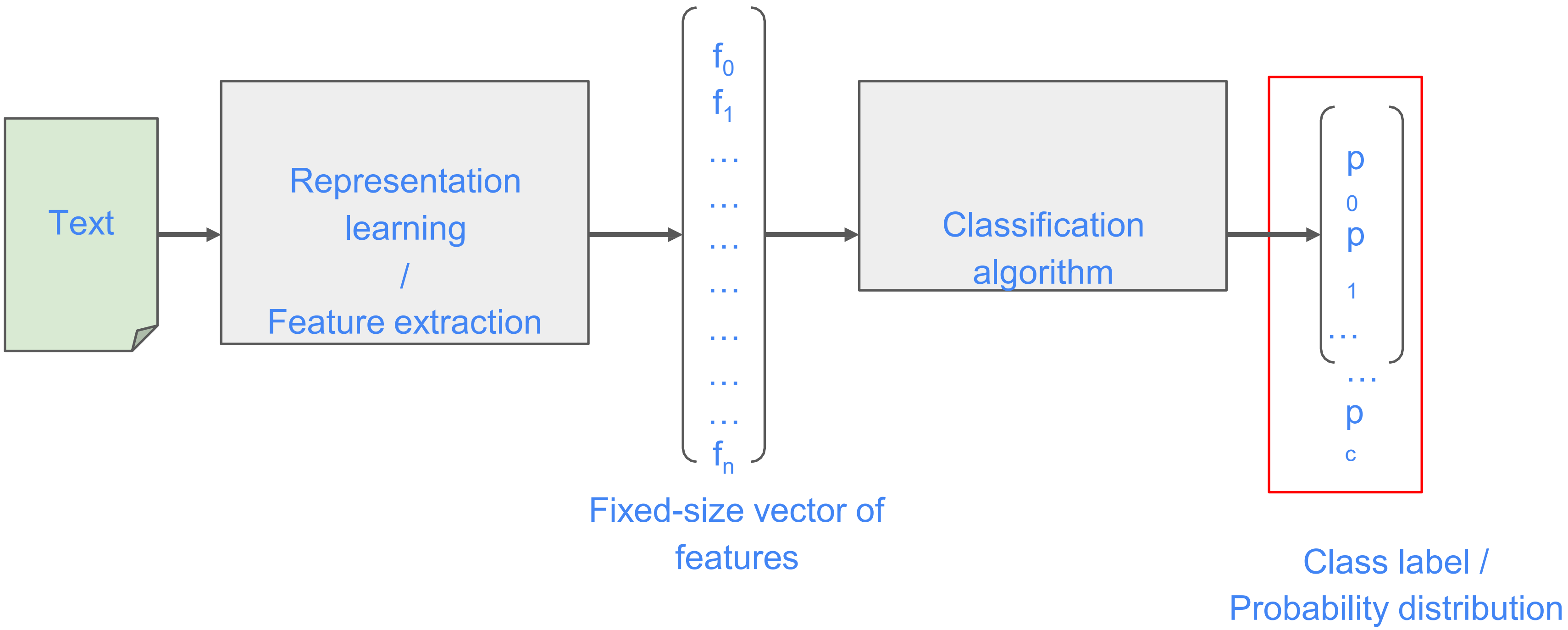


Subjective unigram	Objective unigram
amazing, beautiful, cheap, decent, effective, fantastic, good, happy, impress, jittery, light, madly, nice, outstanding, perfect, quick, responsive, sharp, terrible, ultimate, wonderful.	access, because, chance, default, entire, few, go, half, inside, job, keep, know, last, matter, new, only, past, quality, read, several, text, use, version, was, young.

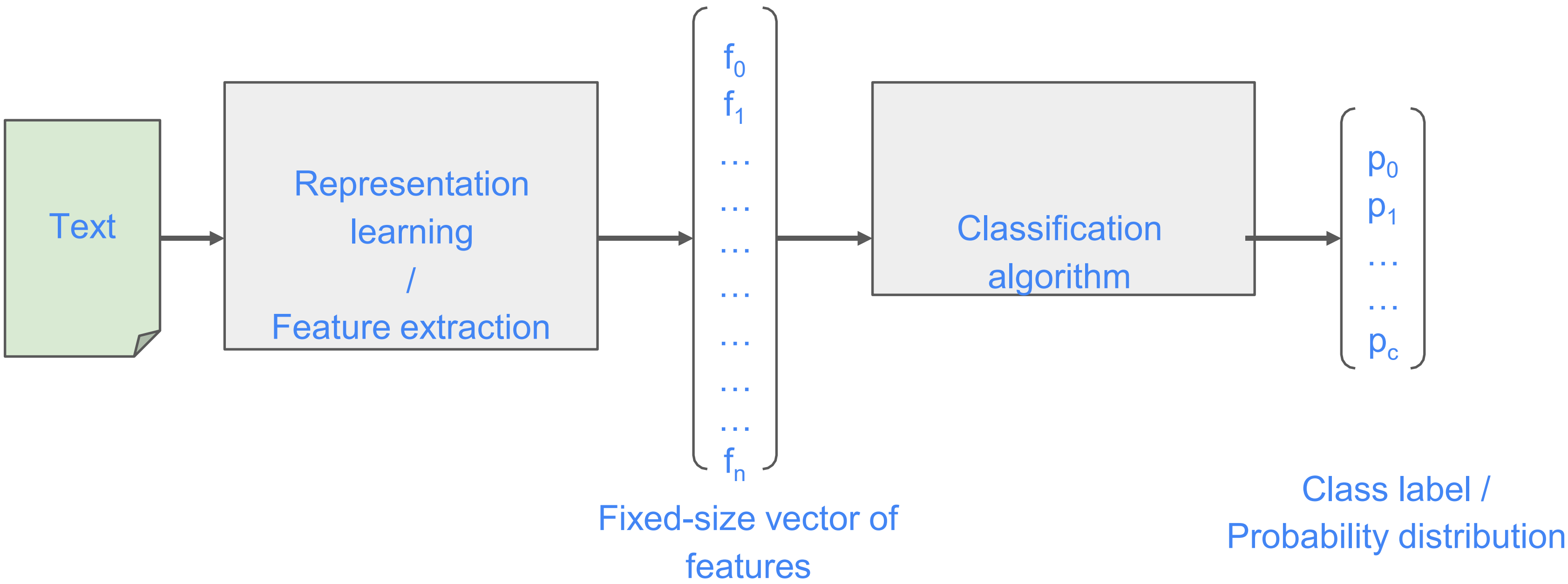
Text classification in general



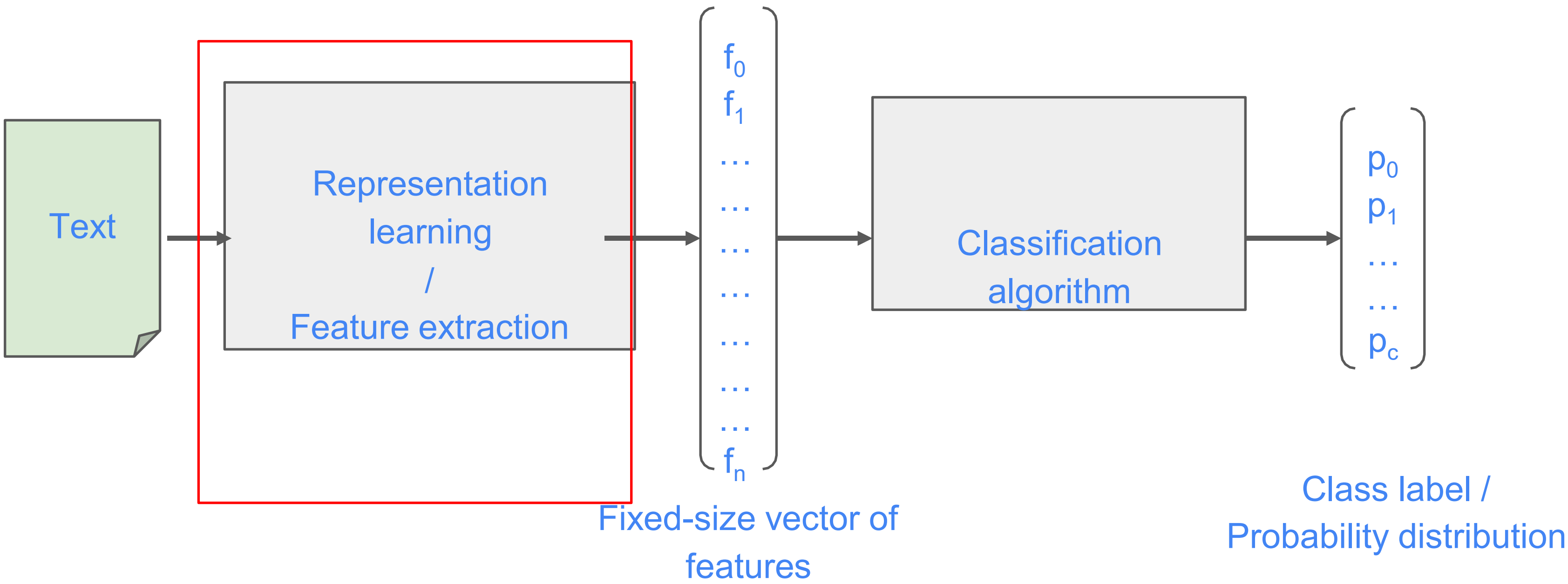
Text classification in general



Text classification in general



Text classification in general



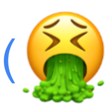
Text representation: feature engineering

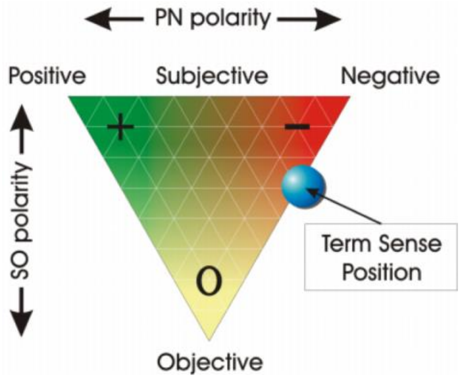
As for many ML tasks, it is possible to generate useful features by hands.

Like what?

Text representation: feature engineering

As for many ML tasks, it is possible to generate useful features by hands.

- General statistics: text length, text length variance,...
- Scores from tagged word lists:
    - Sentiment dictionaries: [SentiWordNet](), [SentiWords](), ...
    - Subjectivity/objectivity dictionaries: [MPQA]()
    - ...
- Syntactic features:
    - POS tags
- Ad-hoc features: e.g. number of emojis ( or )



Sparse text representation: BOW

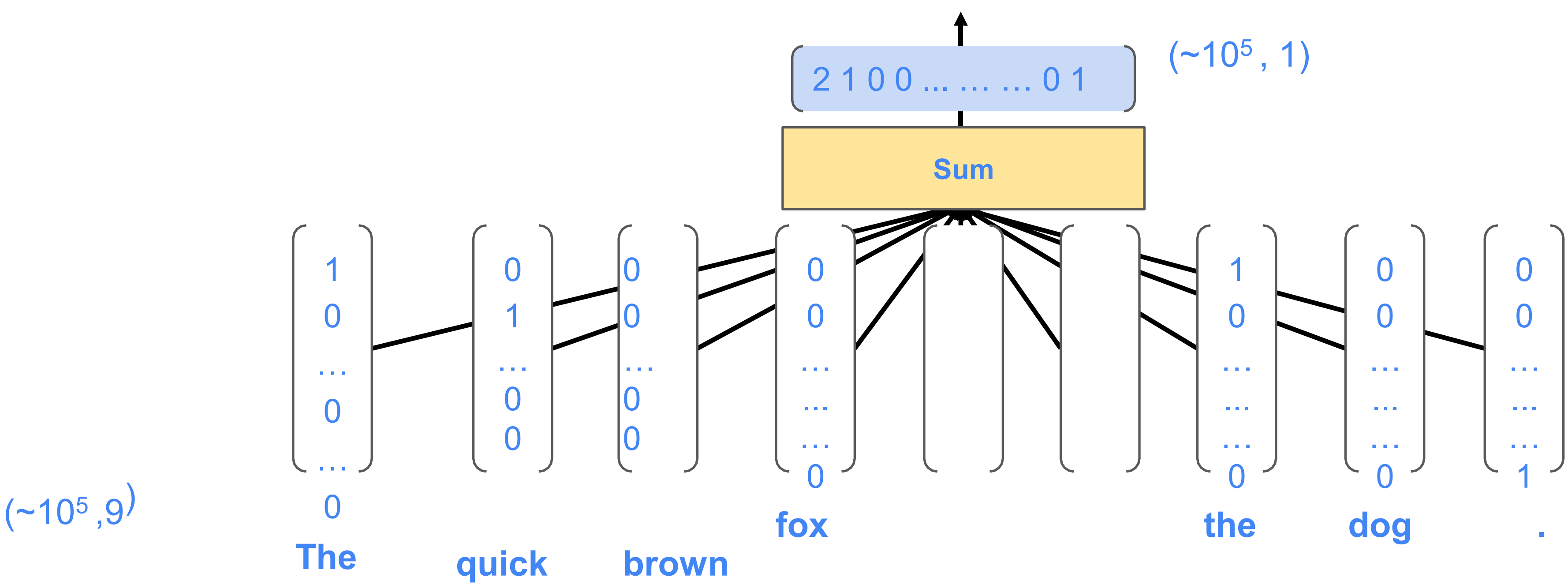
The quick brown fox jumps over the dog .

Sparse text representation: BOW

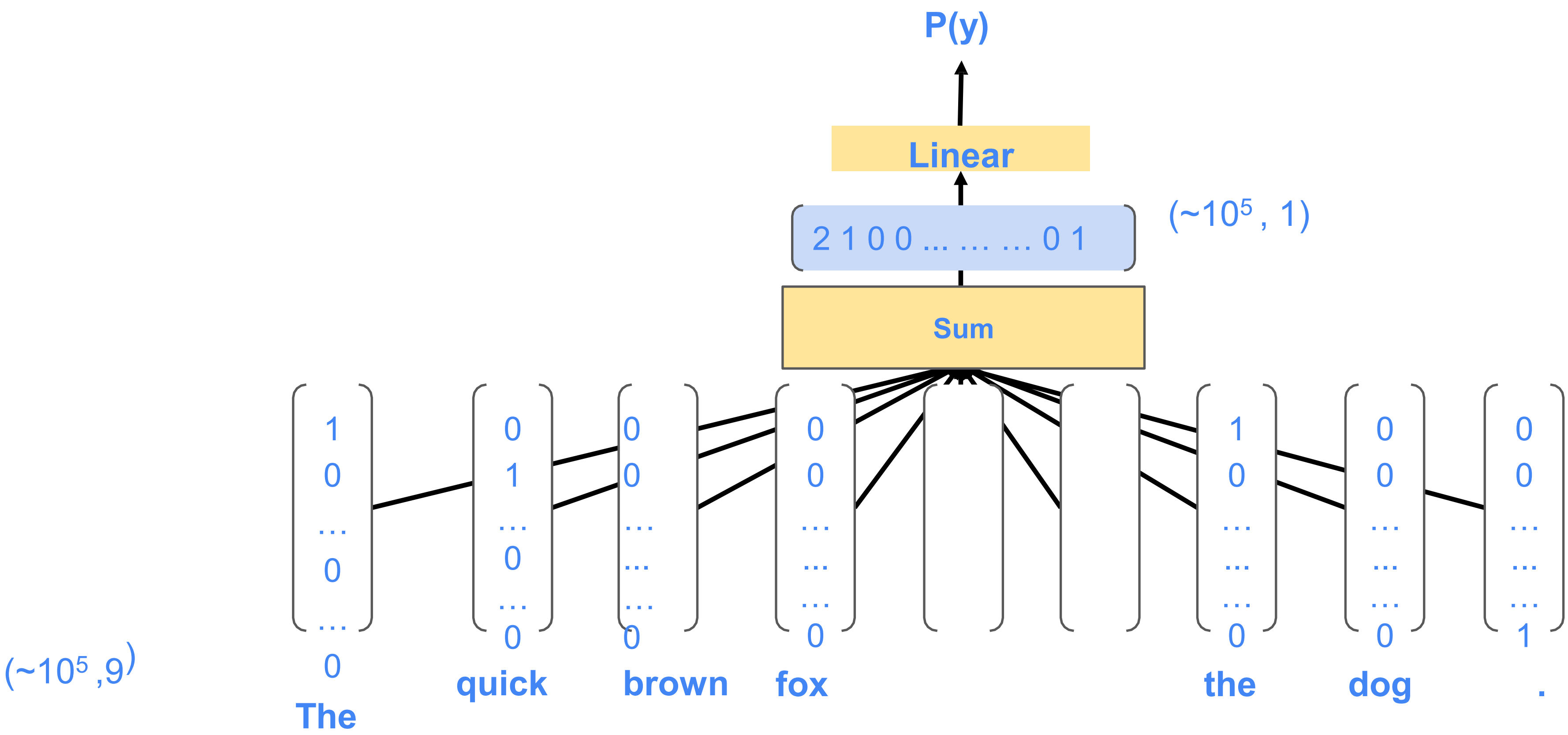
Sparse text representation: BOW

$$
(\sim 10^5, 9) \quad
\begin{array}{ccccccccc}
\begin{pmatrix} 1 \\ 0 \\ \cdots \\ 0 \\ \cdots \\ 0 \end{pmatrix} &
\begin{pmatrix} 0 \\ 1 \\ \cdots \\ 0 \\ \cdots \\ 0 \end{pmatrix} &
\begin{pmatrix} 0 \\ 0 \\ \cdots \\ \cdots \\ \cdots \\ 0 \end{pmatrix} &
\begin{pmatrix} 0 \\ 0 \\ \cdots \\ \cdots \\ \cdots \\ 0 \end{pmatrix} &
\begin{pmatrix} 0 \\ 0 \\ \cdots \\ \cdots \\ \cdots \\ 0 \end{pmatrix} &
\begin{pmatrix} 0 \\ 0 \\ \cdots \\ \cdots \\ \cdots \\ 0 \end{pmatrix} &
\begin{pmatrix} 1 \\ 0 \\ \cdots \\ \cdots \\ \cdots \\ 0 \end{pmatrix} &
\begin{pmatrix} 0 \\ 0 \\ \cdots \\ \cdots \\ \cdots \\ 0 \end{pmatrix} &
\begin{pmatrix} 0 \\ 0 \\ \cdots \\ \cdots \\ \cdots \\ 1 \end{pmatrix} \\
\textbf{The} & \textbf{quick} & \textbf{brown} & \textbf{fox} & \textbf{jumps} & \textbf{over} & \textbf{the} & \textbf{dog} & \textbf{.}
\end{array}
$$

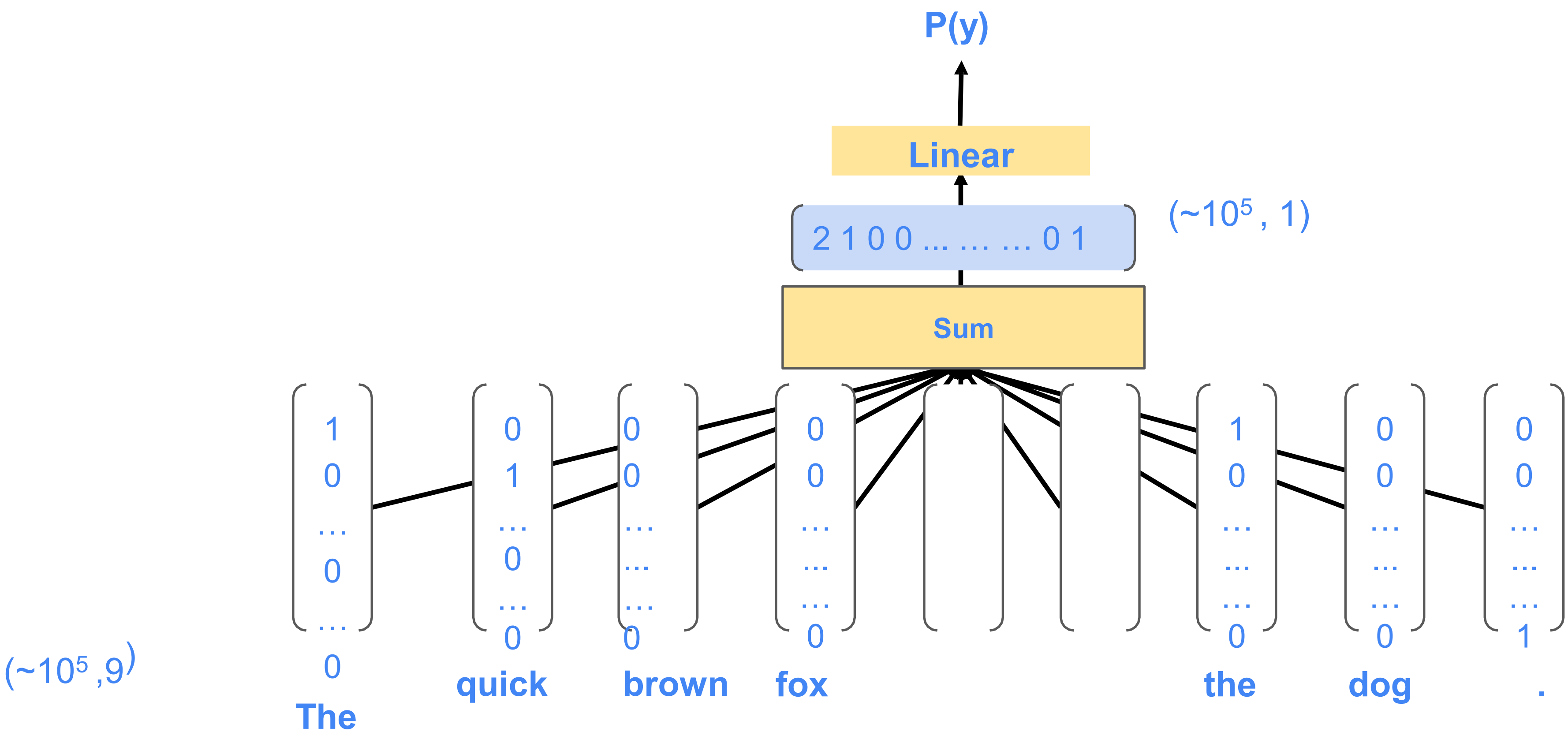
Sparse text representation: BOW



Sparse text representation: BOW



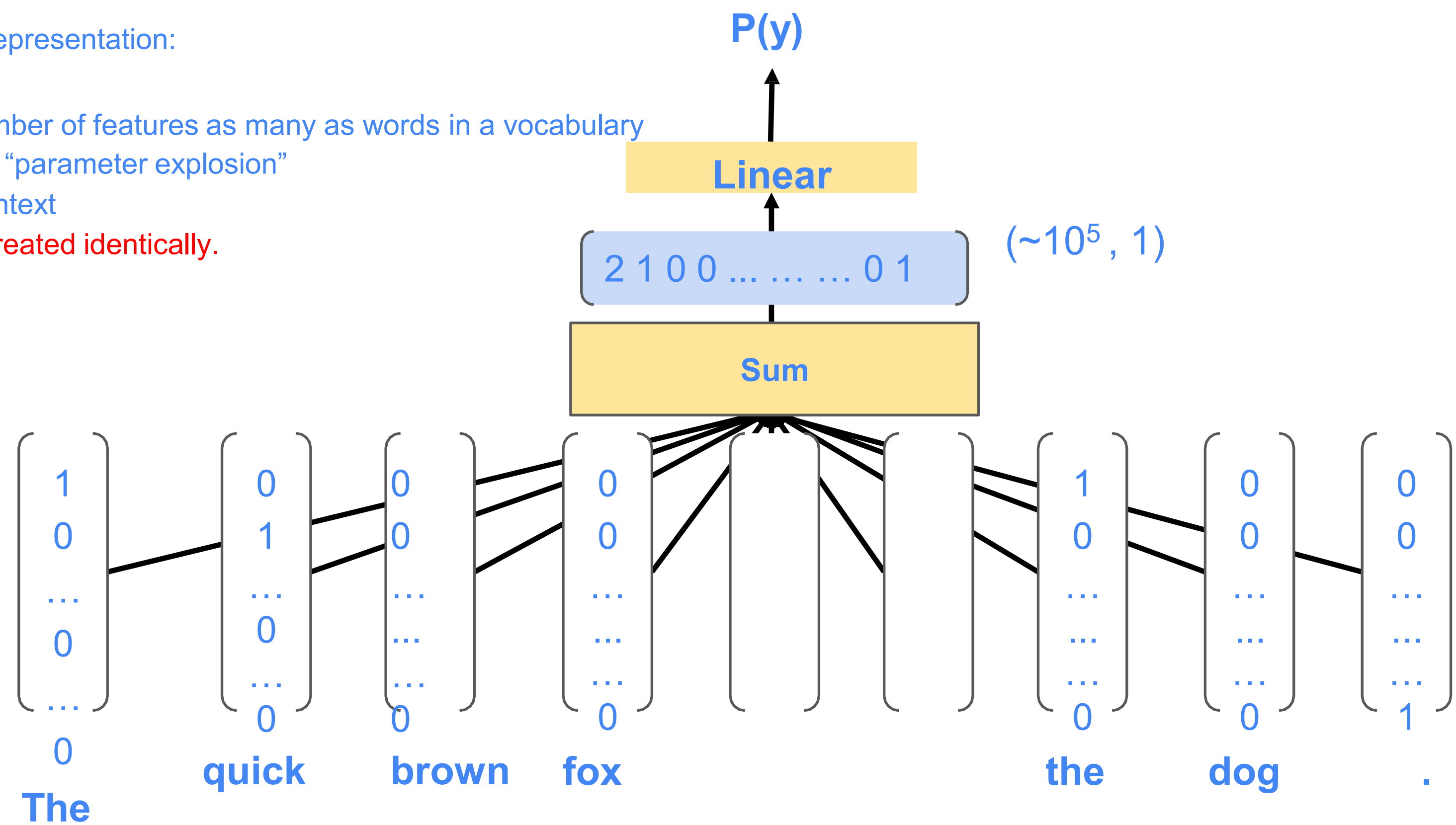
Sparse text representation: BOW



Sparse text representation: BOW

Problems with BOW representation:

1. The number of features as many as words in a vocabulary (~10^4-10^5). So called “parameter explosion”
2. Absence of context
3. All words are treated identically.

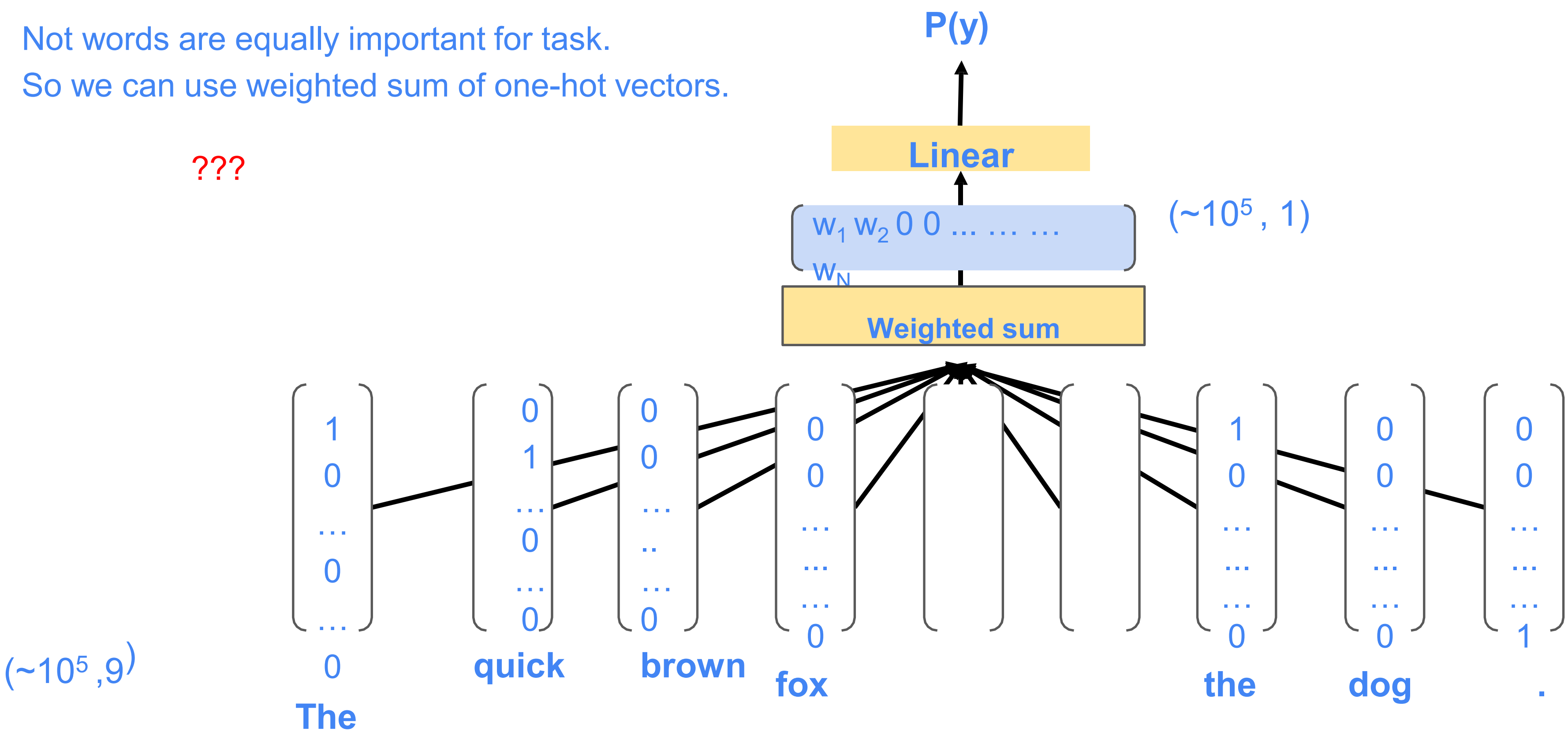


(~10^5 ,9)

Weighting techniques for BOW

Not words are equally important for task.
So we can use weighted sum of one-hot vectors.

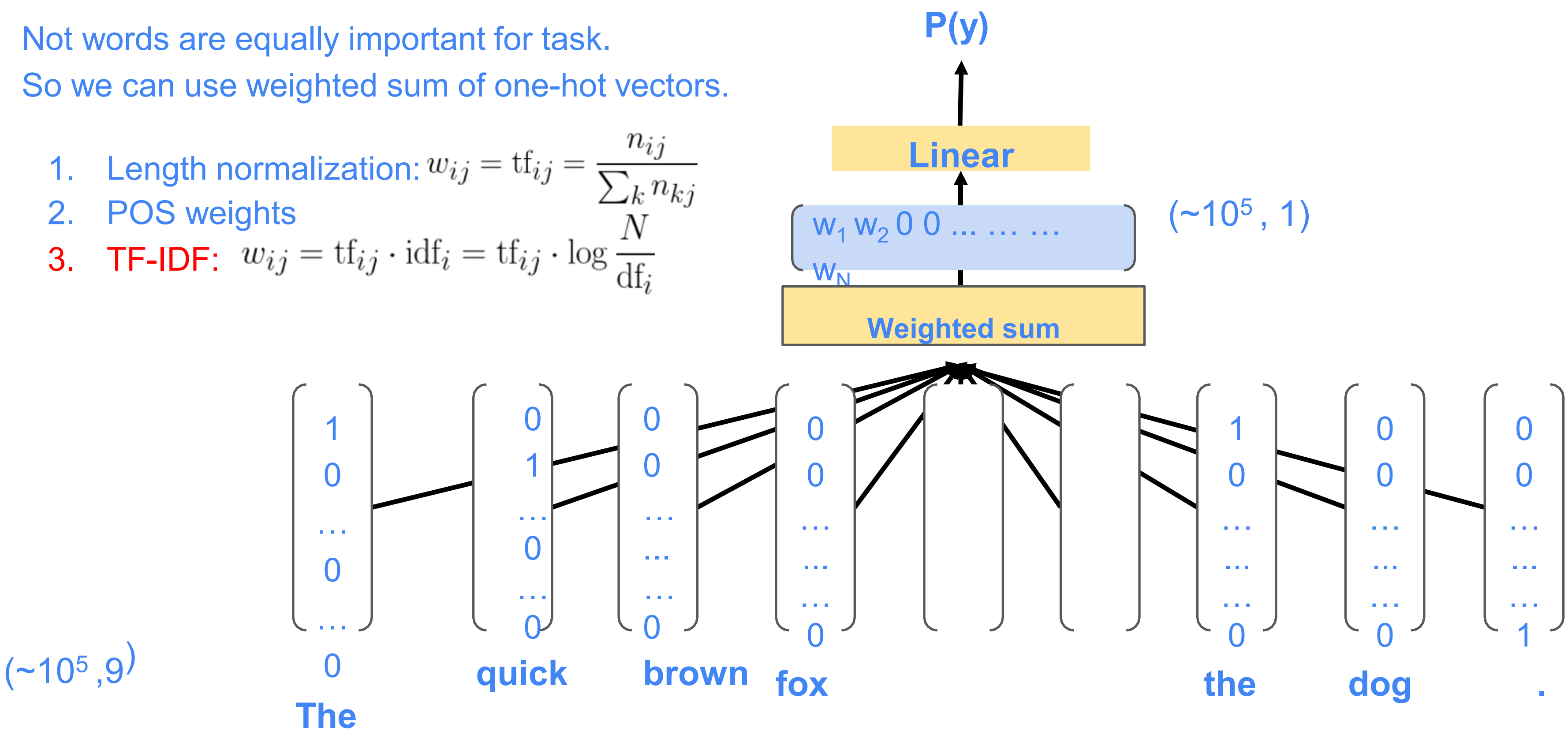
???



Weighting techniques for BOW

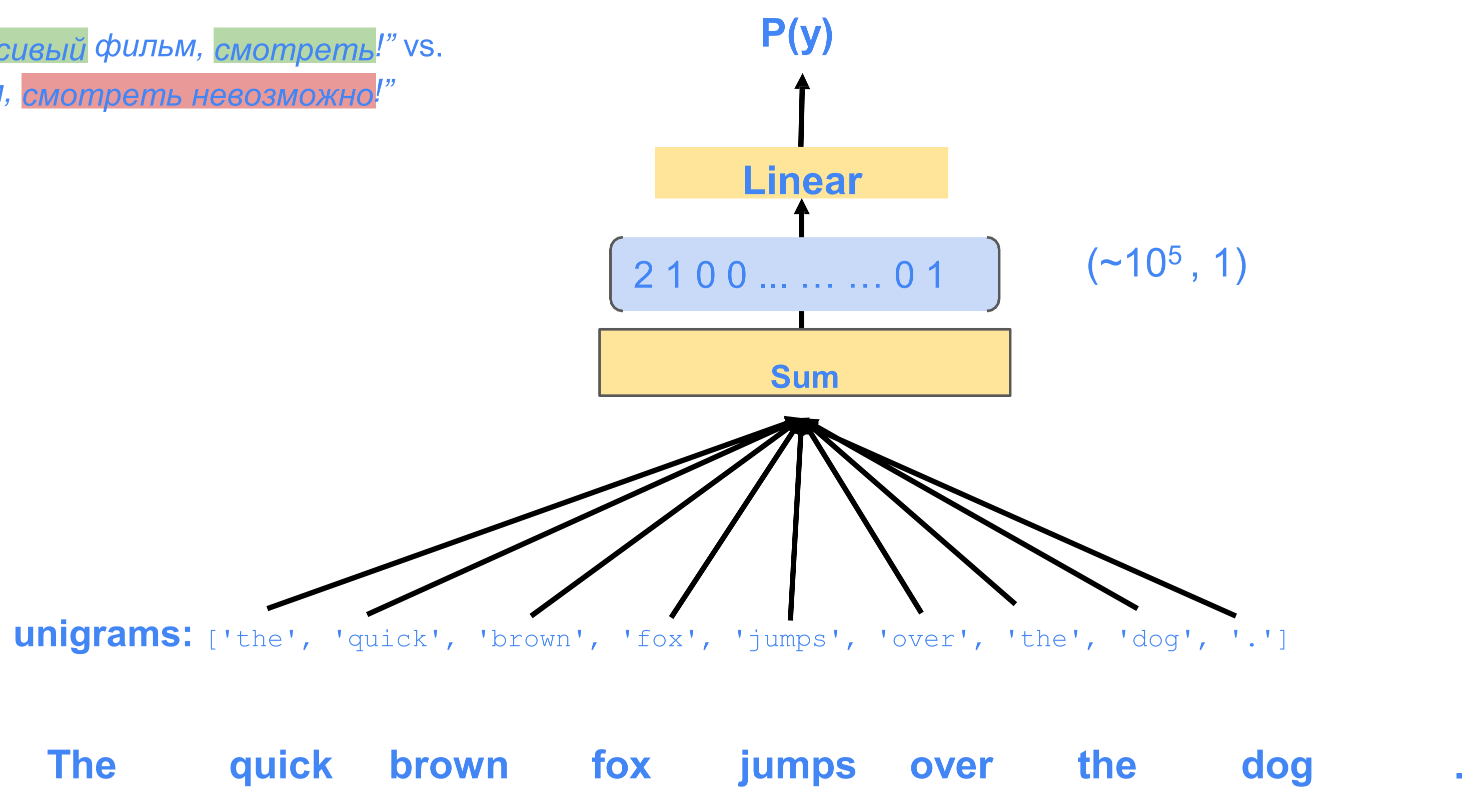
Not words are equally important for task.
So we can use weighted sum of one-hot vectors.

1. Length normalization: $w_{ij} = \mathrm{tf}_{ij} = \frac{n_{ij}}{\sum_k n_{kj}}$
2. POS weights
3. TF-IDF: $w_{ij} = \mathrm{tf}_{ij} \cdot \mathrm{idf}_i = \mathrm{tf}_{ij} \cdot \log \frac{N}{\mathrm{df}_i}$



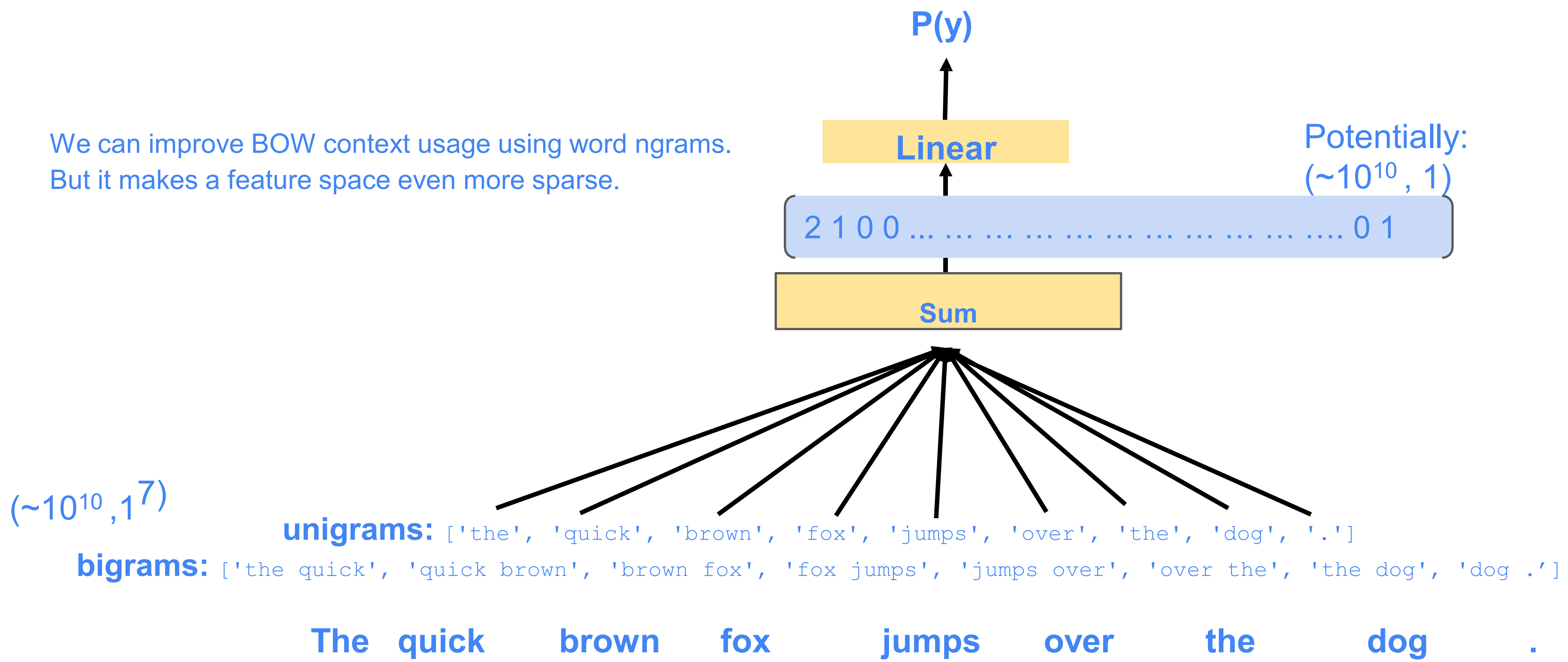
Context importance

“Невозможно красивый фильм, смотреть!” vs.
“Красивый фильм, смотреть невозможно!”



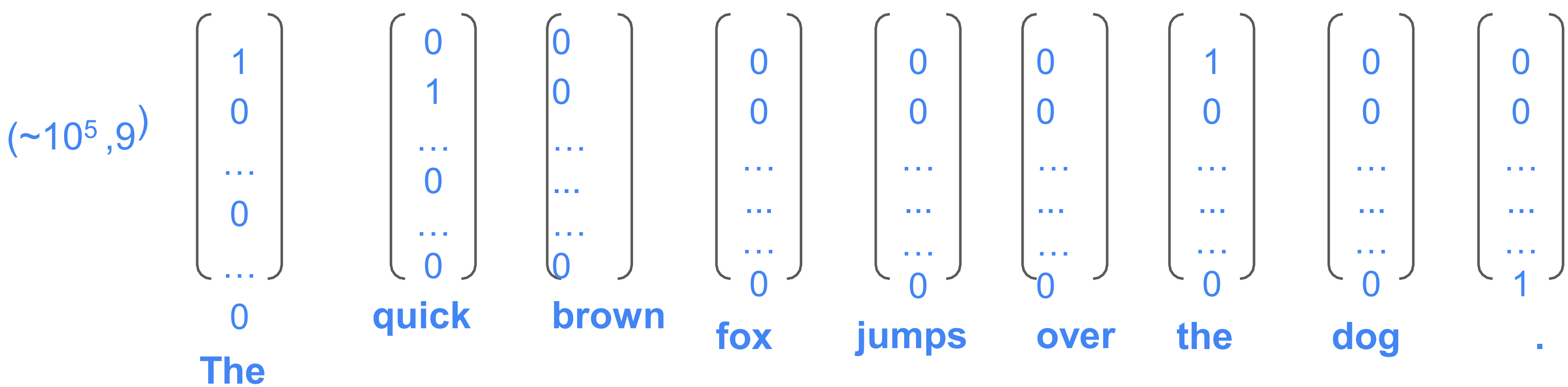
Context importance

We can improve BOW context usage using word ngrams.
But it makes a feature space even more sparse.



Dense text representation: NBOW

Instead of sparse one-hot encoding we can you use pre- trained word embeddings. It is so-called “neural bag-of- words”**.**



Dense text representation: NBOW

Instead of sparse one-hot encoding we can you use pre- trained word embeddings. It is so-called “neural bag-of- words”**.**

$(\sim 10^3, 9)$

The	quick	brown	fox	jumps	over	the	dog	.
0.3	0.7	0.7	8.2	0.0	0.6	0.3	8.0	0.0
3.5	9.3	4.4	1.5	3.4	1.8	3.5	9.8	3.4
…	…	…	…	…	…	…	…	…
4.1	3.9	7.8	5.1	6.1	3.1	4.1	5.1	6.1
…	…	…	…	…	…	…	…	…
0.4	0.3	1.9	0.4	2.7	1.9	0.4	1.8	2.7
0.1	0.9	8.8	0.5	9.1	8.8	0.1	6.1	9.1

Dense text representation: NBOW

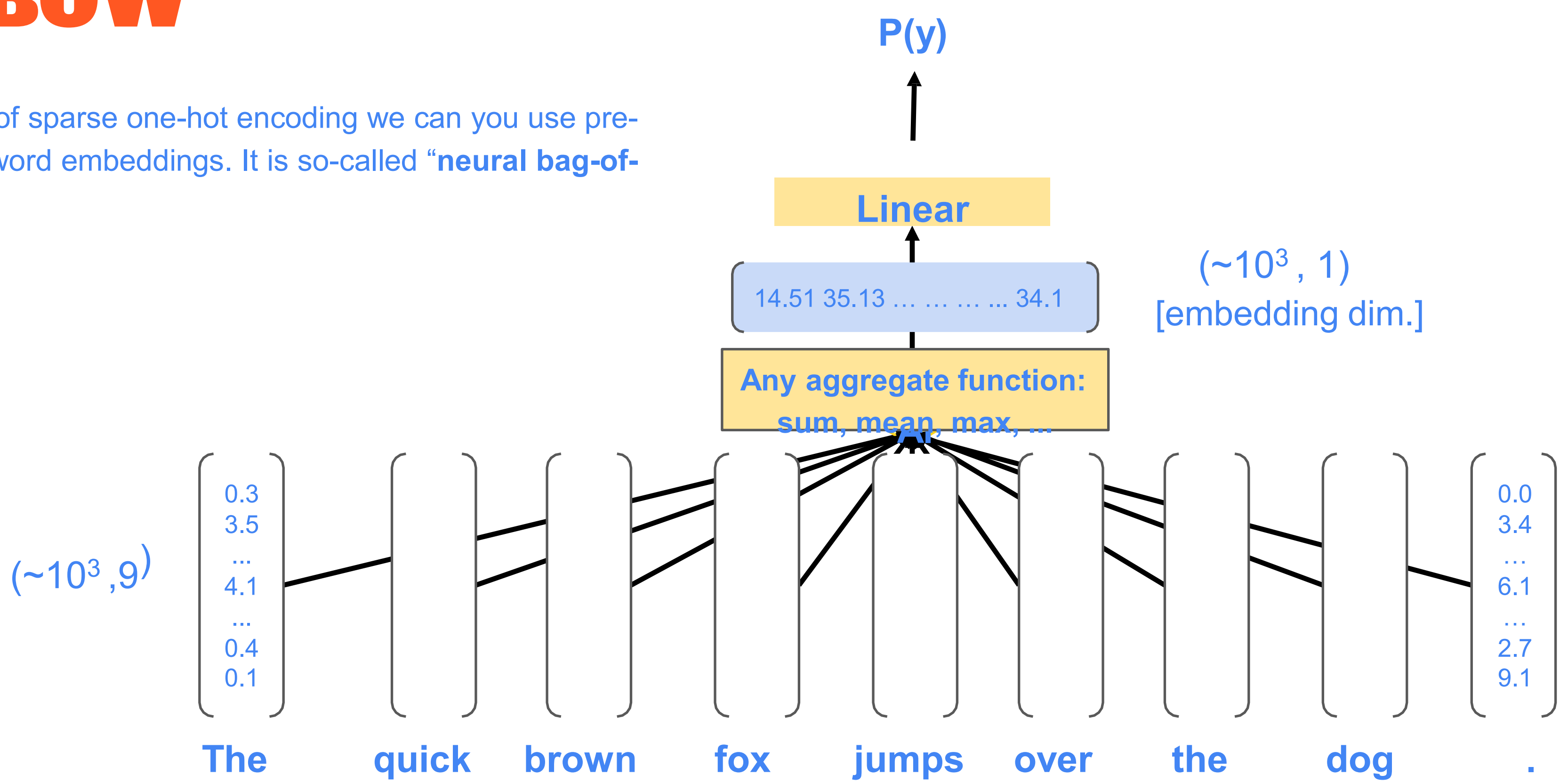
Instead of sparse one-hot encoding we can you use pre- trained word embeddings. It is so-called “neural bag-of- words”**.**

$(\sim 10^3, 9)$

0.3	0.7	0.7	8.2	0.0	0.6	0.3	8.0	0.0
3.5	9.3	4.4	1.5	3.4	1.8	3.5	9.8	3.4
...	…	…	…	…	…	...	…	…
4.1	3.9	7.8	5.1	6.1	3.1	4.1	5.1	6.1
...	…	…	…	…	…	...	…	…
0.4	0.3	1.9	0.4	2.7	1.9	0.4	1.8	2.7
0.1	0.9	8.8	0.5	9.1	8.8	0.1	6.1	9.1
The	**quick**	**brown**	**fox**	**jumps**	**over**	**the**	**dog**	**.**

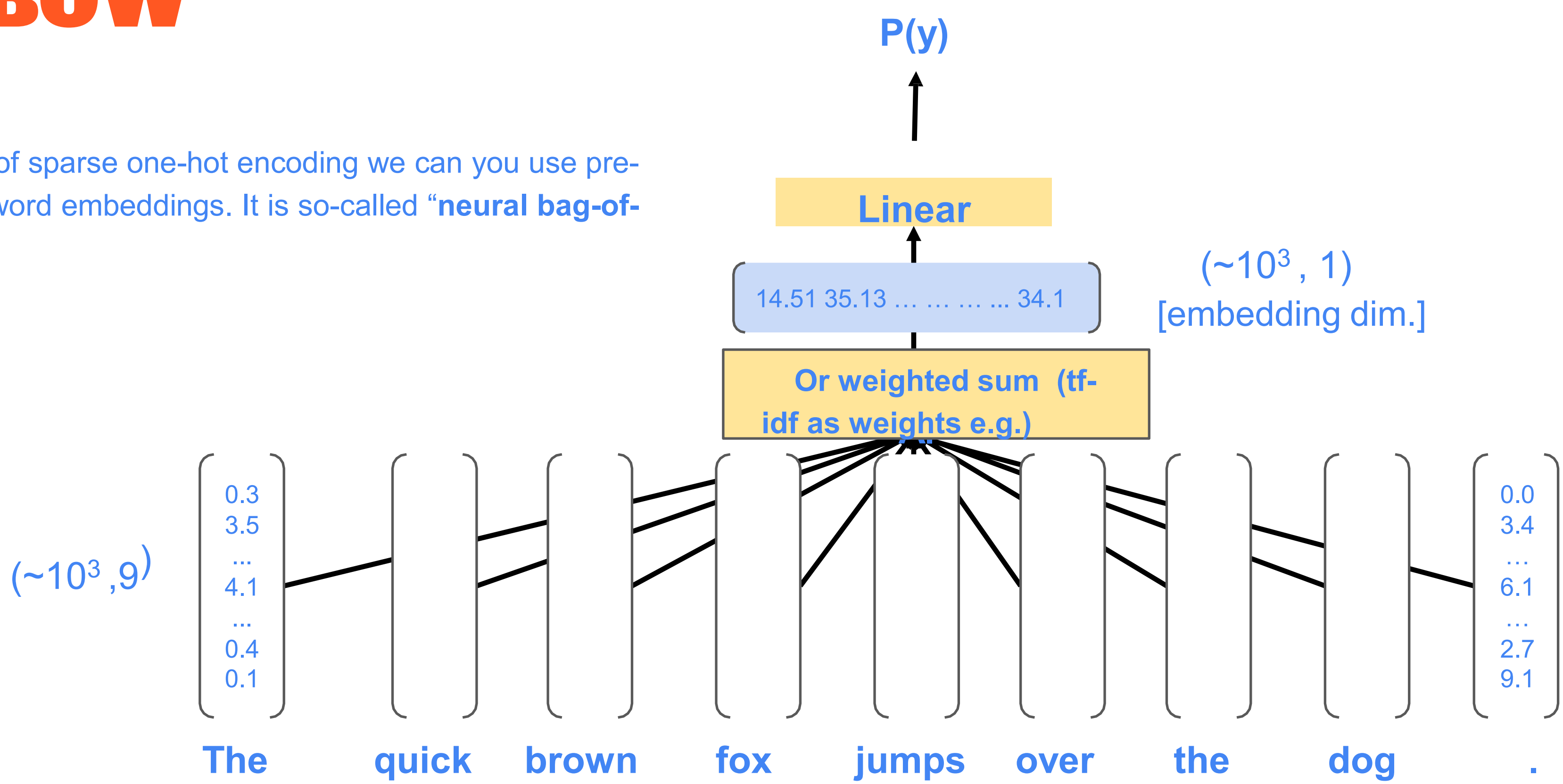
Dense text representation: NBOW

Instead of sparse one-hot encoding we can you use pre-trained word embeddings. It is so-called “**neural bag-of-words**”.



Dense text representation: NBOW

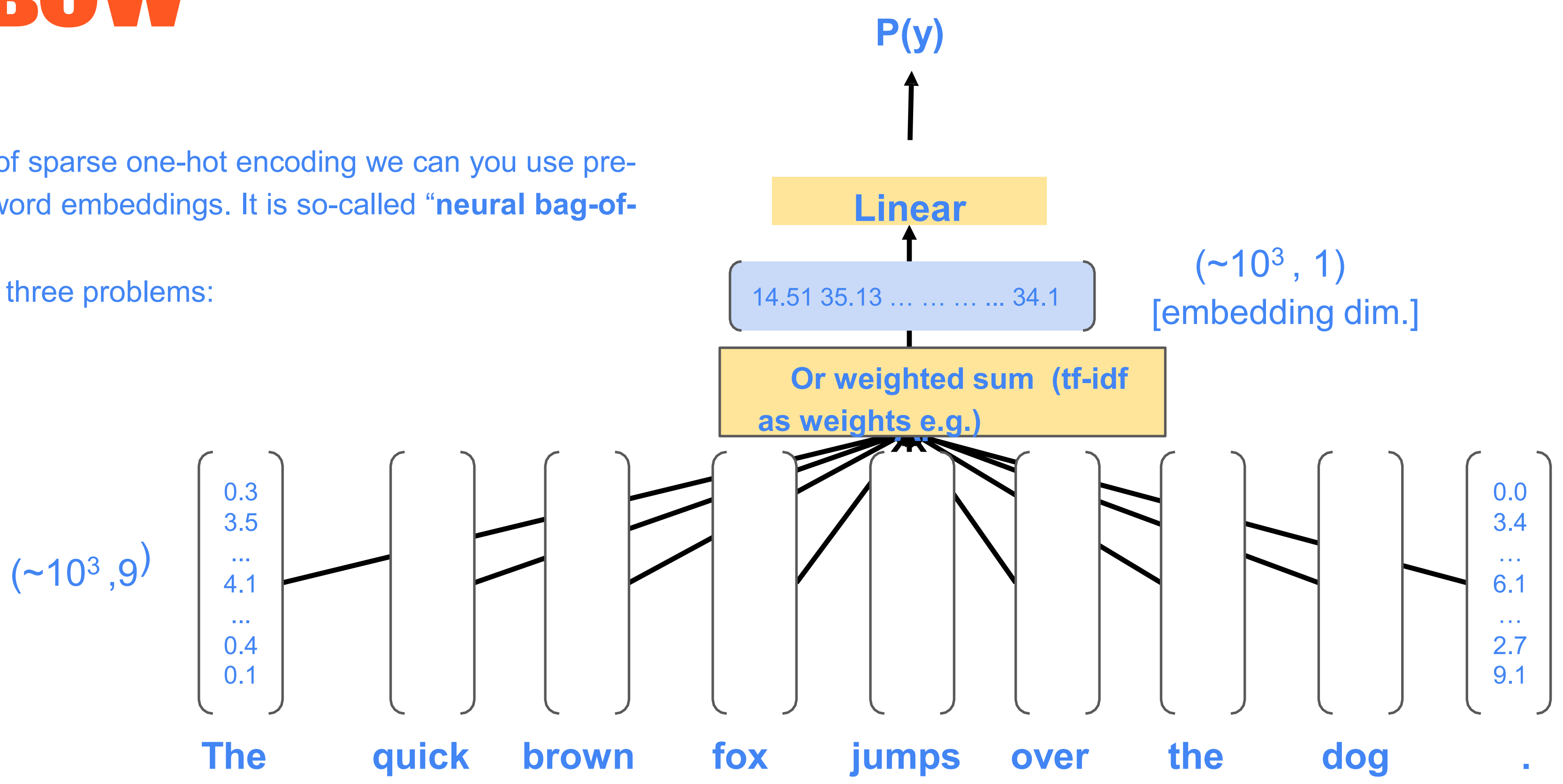
Instead of sparse one-hot encoding we can you use pre-trained word embeddings. It is so-called “**neural bag-of-words**”.



Dense text representation: NBOW

Instead of sparse one-hot encoding we can you use pre-trained word embeddings. It is so-called “**neural bag-of-words**”.

It solves three problems:

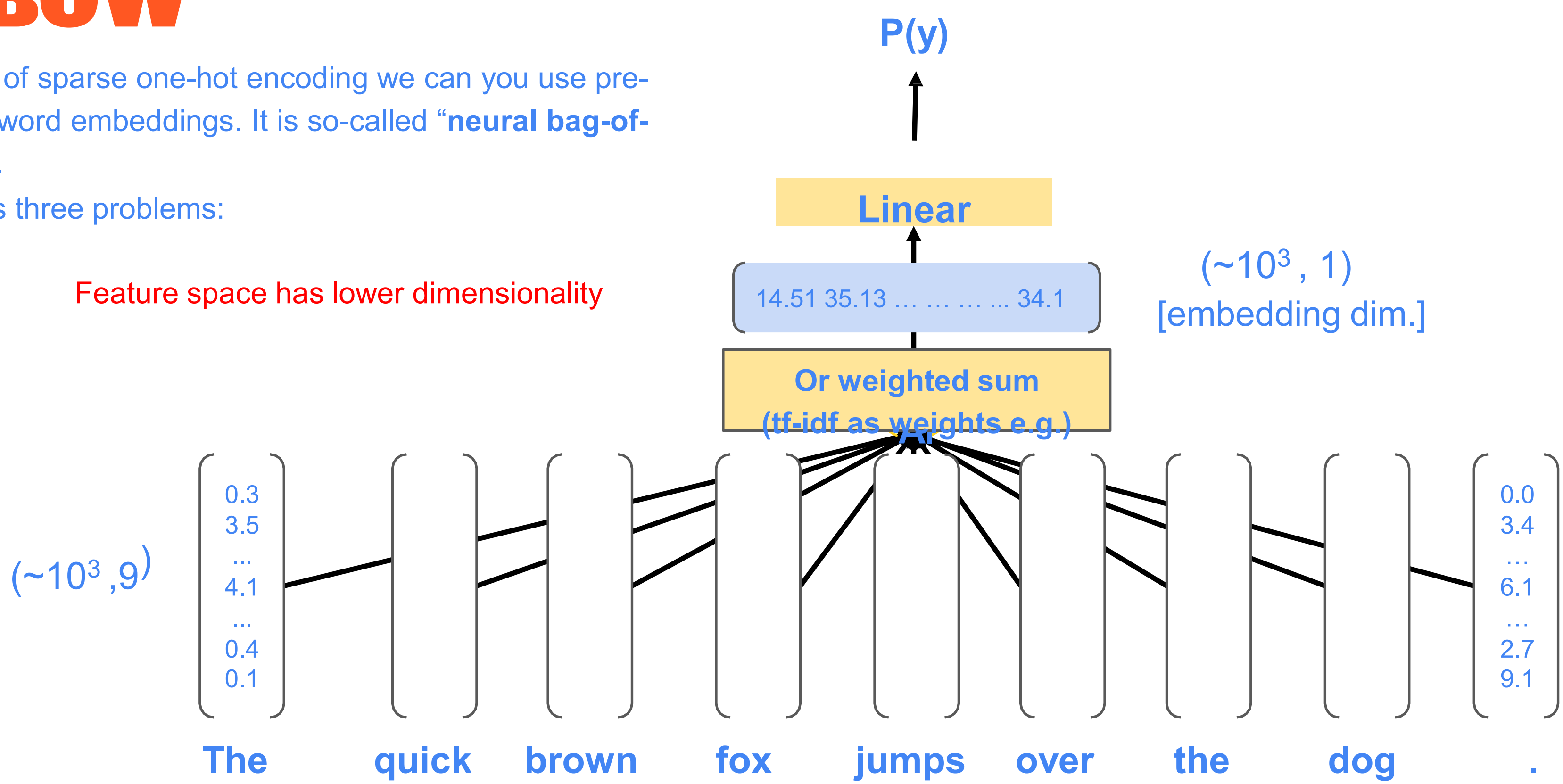


Dense text representation: NBOW

Instead of sparse one-hot encoding we can you use pre-trained word embeddings. It is so-called “**neural bag-of-words**”.

It solves three problems:

1. Feature space has lower dimensionality

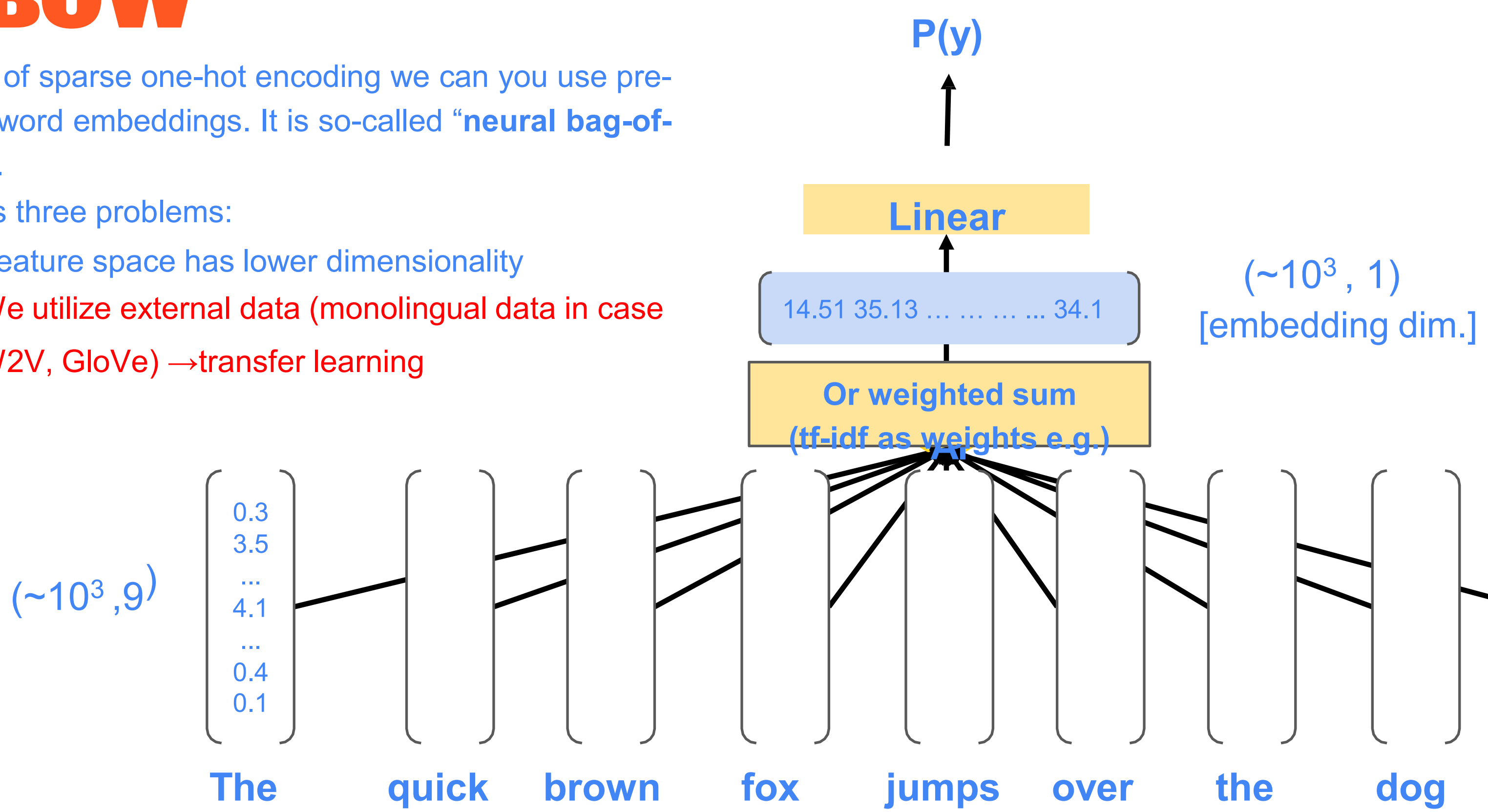


Dense text representation: NBOW

Instead of sparse one-hot encoding we can you use pre-trained word embeddings. It is so-called “**neural bag-of-words**”.

It solves three problems:

1. Feature space has lower dimensionality
2. We utilize external data (monolingual data in case W2V, GloVe) →transfer learning

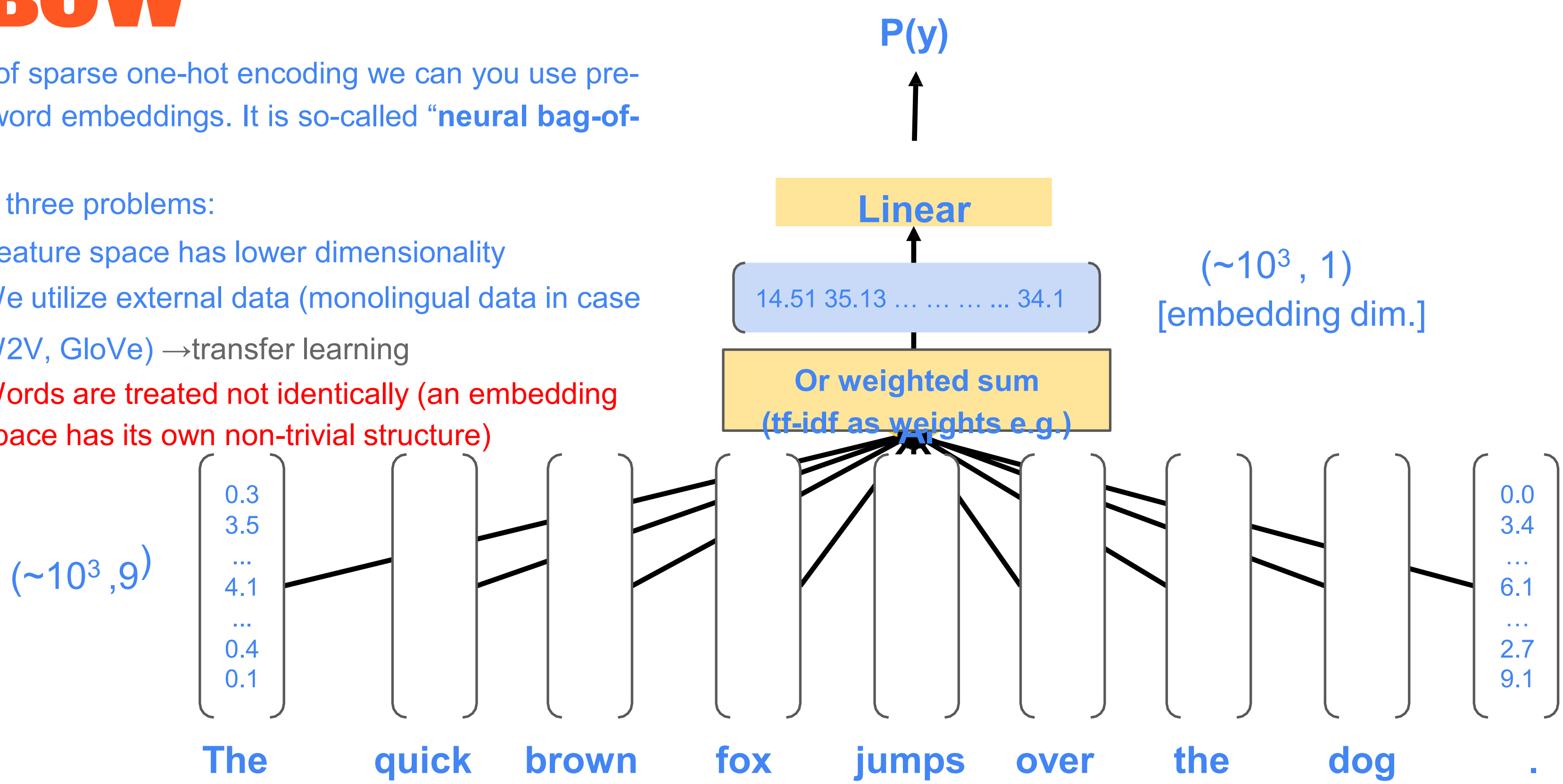


Dense text representation: NBOW

Instead of sparse one-hot encoding we can you use pre-trained word embeddings. It is so-called “**neural bag-of-words**”.

It solves three problems:

1. Feature space has lower dimensionality
2. We utilize external data (monolingual data in case W2V, GloVe) →transfer learning
3. Words are treated not identically (an embedding space has its own non-trivial structure)



Dense text representation: NBOW

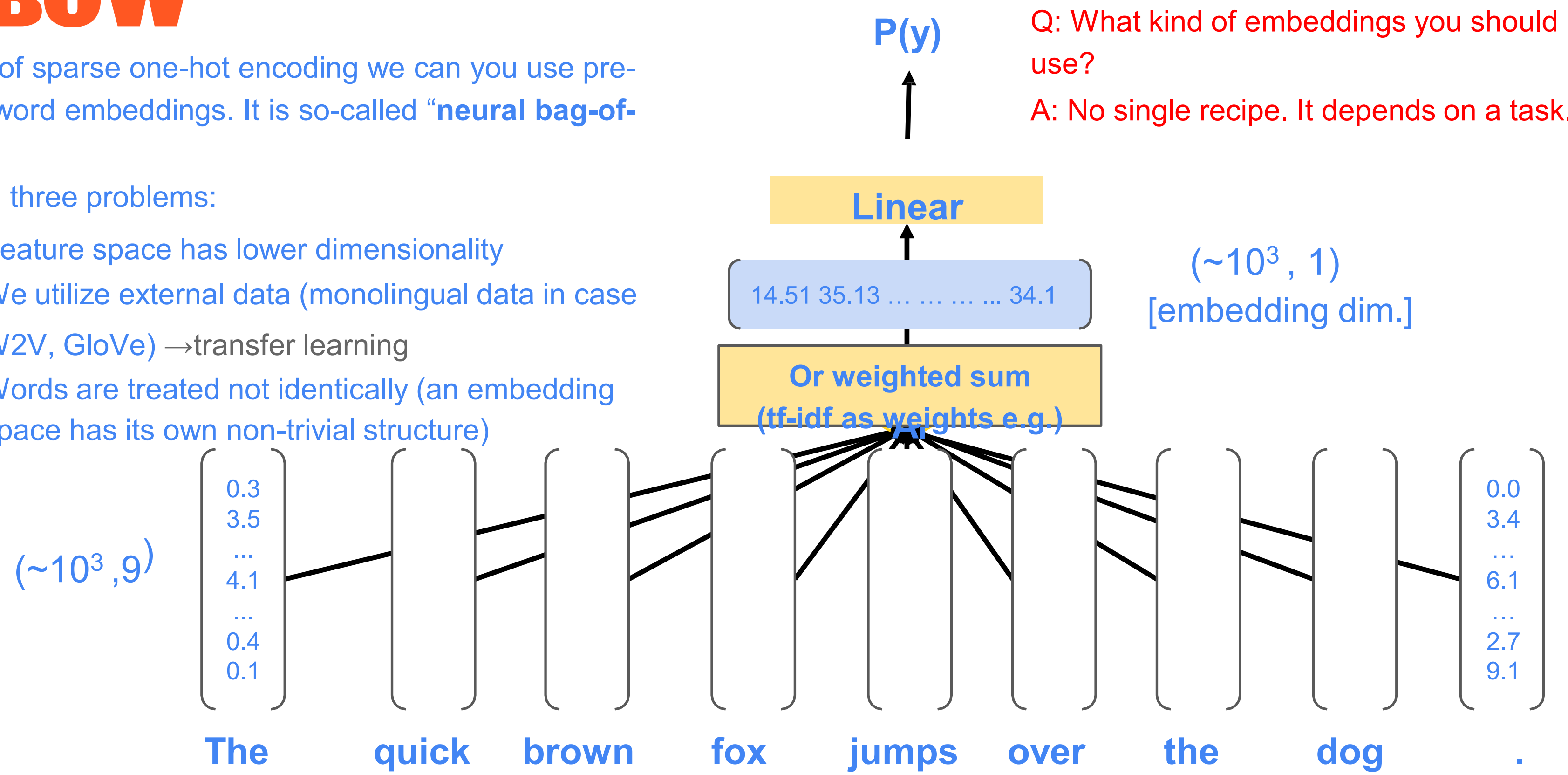
Instead of sparse one-hot encoding we can you use pre-trained word embeddings. It is so-called “**neural bag-of-words**”.

It solves three problems:

1. Feature space has lower dimensionality
2. We utilize external data (monolingual data in case W2V, GloVe) →transfer learning
3. Words are treated not identically (an embedding space has its own non-trivial structure)

Q: What kind of embeddings you should use?

A: No single recipe. It depends on a task.



Dense text representation: NBOW

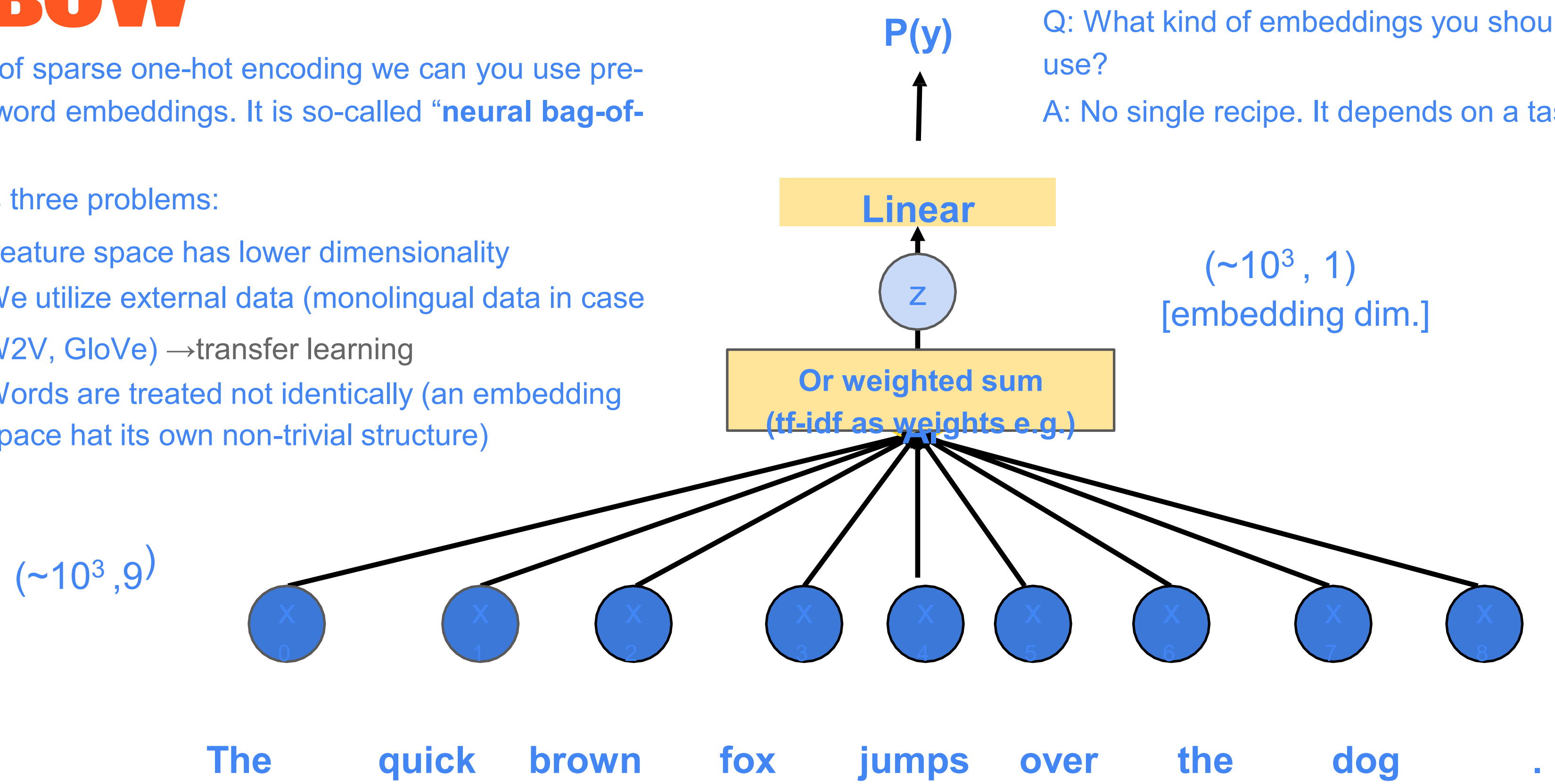
Instead of sparse one-hot encoding we can you use pre-trained word embeddings. It is so-called “**neural bag-of-words**”.

It solves three problems:

1. Feature space has lower dimensionality
2. We utilize external data (monolingual data in case W2V, GloVe) →transfer learning
3. Words are treated not identically (an embedding space hat its own non-trivial structure)

Q: What kind of embeddings you should use?

A: No single recipe. It depends on a task.



BOW and NBOW: the shared problems

1. The importance weights for the word vectors aren't defined fully.
2. The only way to use context for these models is to utilize word ngrams.

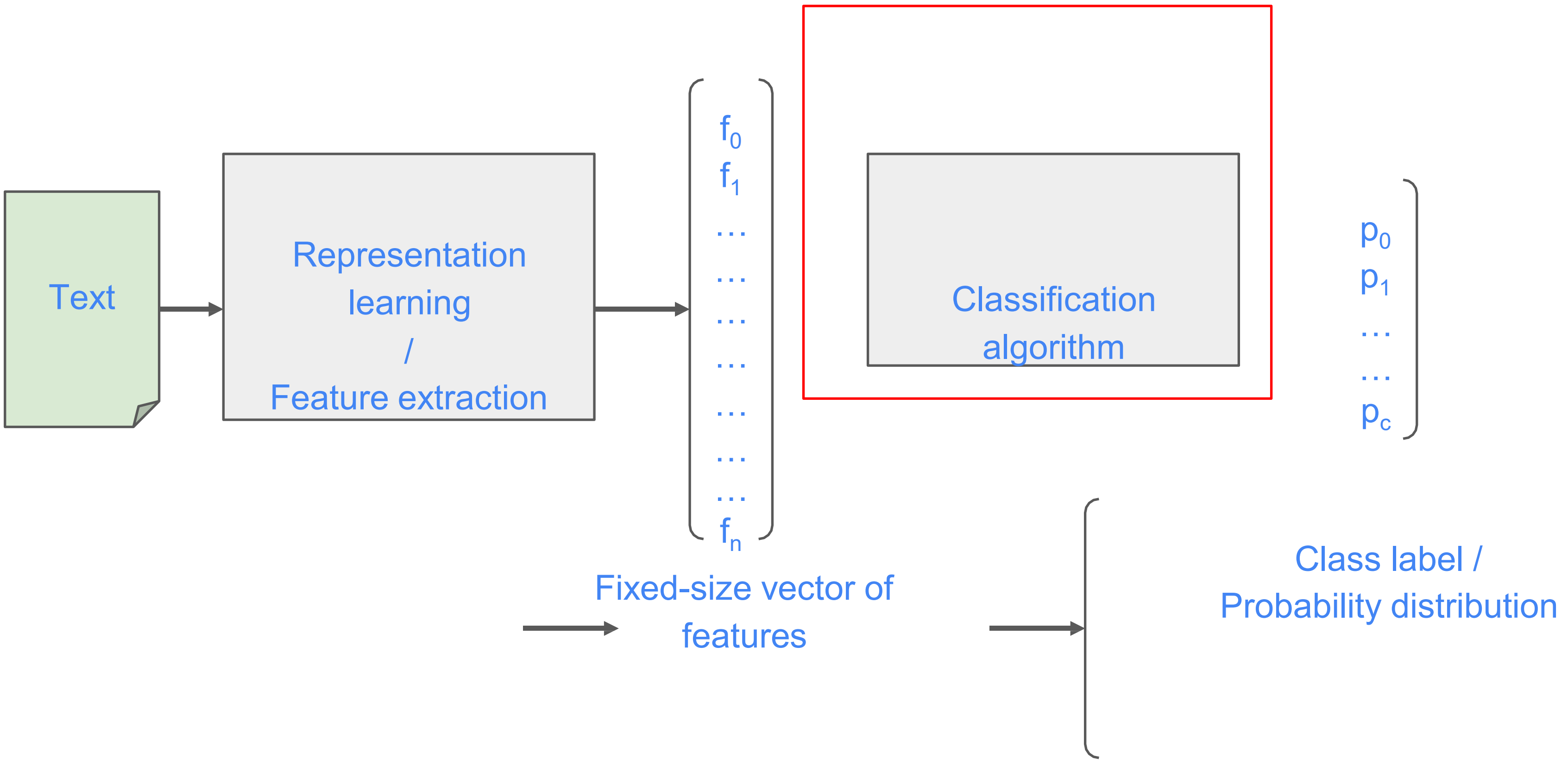
Hmm...

BOW and NBOW: the shared problems

1. The importance weights for the word vectors aren’t defined fully.
2. The only way to use context for these models is to utilize word ngrams.

We can use a learnable aggregation function to overcome the difficulties.
The learnable function is a neural network (the universal approximator)

Text classification in general



Classification Algorithms

Let’s explore how to use Naïves bayes to solve this task

Naïve Bayes Intuition

Simple ("naïve") classification method based on Bayes rule

Relies on very simple representation of document

- Bag of words

The bag of words representation

$\gamma($

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun… It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.

$)=c$

The bag of words representation

$\gamma($

I **love** this movie! It's **sweet**, but with **satirical** humor. The dialogue is **great** and the adventure scenes are **fun**… It manages to be **whimsical** and **romantic** while **laughing** at the conventions of the fairy tale genre. I would **recommend** it to just about anyone. I've seen it **several** times, and I'm always **happy** to see it **again** whenever I have a friend who hasn't seen it yet.

$)=c$

The bag of words representation: using a subset of words

$\gamma($

```
x love xxxxxxxxxxxxxxxx sweet
xxxxxxx satirical xxxxxxxxxx
xxxxxxxxxxx great xxxxxxx
xxxxxxxxxxxxxxxxxxx fun  xxxx
xxxxxxxxxxxxx whimsical xxxx
romantic xxxx  laughing
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxx recommend xxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xx several xxxxxxxxxxxxxxxxx
xxxxx  happy xxxxxxxxx again
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxx
```

$) = c$

The bag of words representation

$\gamma($

great	2
love	2
recommend	1
laugh	1
happy	1
...	...

$)=c$

Bayes' Rule Applied to Documents and Classes

- For a document *d* and a class *c*

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

Naïve Bayes Classifier (I)

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(c \mid d)$$

MAP is "maximum a posteriori" = most likely class

$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(d \mid c)P(c)}{P(d)}$$

Bayes Rule

$$= \underset{c \in C}{\operatorname{argmax}} P(d \mid c)P(c)$$

Dropping the denominator

Likelihood

Prior probability

Naïve Bayes Classifier (II)

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(d \mid c) P(c)$$

$$= \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

Document d represented as features x1..xn

Naïve Bayes Classifier (III)

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

O(|X|^n•|C|) parameters

Could only be estimated if a very, very large number of training examples was available.

How often does this class occur?

We can just count the relative frequencies in a corpus

Multinomial Naïve Bayes Independence Assumptions

$$P(x_1, x_2, \ldots, x_n \mid c)$$

Bag of Words assumption: Assume position doesn't matter

Conditional Independence: Assume the feature probabilities $P(x_i|c_j)$ are independent given the class c.

$$P(x_1, \ldots, x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet \ldots \bullet P(x_n \mid c)$$

Multinomial Naïve Bayes Classifier

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \cdots, x_n \mid c) P(c)$$

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c_j) \prod_{x \in X} P(x \mid c)$$

Applying Multinomial Naive Bayes Classifiers to Text Classification

positions $\leftarrow$ all word positions in test document

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Problems with multiplying lots of probs

There's a problem with this:

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Multiplying lots of probabilities can result in floating-point underflow!

.0006 * .0007 * .0009 * .01 * .5 * .000008….

Idea: Use logs, because log(*ab*) = log(*a*) + log(*b*)

We'll sum logs of probabilities instead of multiplying probabilities!

We actually do everything in log space

Instead of this: $$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

This: $$c_{\text{NB}} = \underset{c_j \in C}{\operatorname{argmax}} \left[\log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i | c_j) \right]$$

Notes:

1) Taking log doesn't change the ranking of classes!

    The class with highest probability also has highest log probability!

2) It's a linear model:

    Just a max of a sum of weights: a **linear** function of the inputs

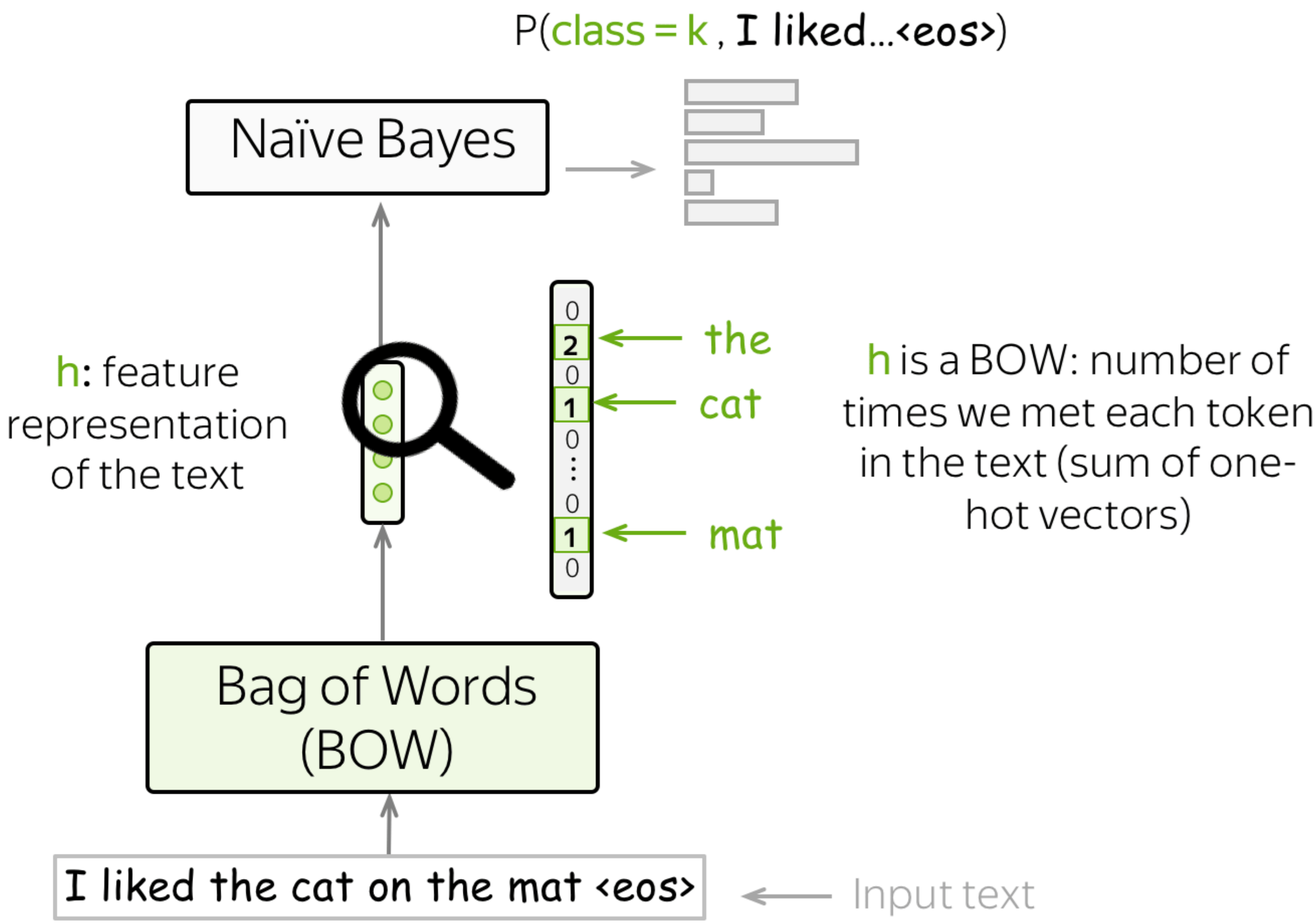
    So naive bayes is a **linear classifier**

Laplace (add-1) smoothing for Naïve Bayes

$$\hat{P}(w_i \mid c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} \left(count(w, c) + 1 \right)}$$

$$= \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c) \right) + |V|}$$

Final Framework



A Worked Example

$$\hat{P}(c) = \frac{N_c}{N}$$

$$\hat{P}(w \mid c) = \frac{count(w,c)+1}{count(c)+|V|}$$

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Priors:

$P(c) = \frac{3}{4}$

$P(j) = \frac{1}{4}$

Conditional Probabilities:

P(Chinese|*c*) = (5+1) / (8+6) = 6/14 = 3/7

P(Tokyo|*c*) = (0+1) / (8+6) = 1/14

P(Japan|*c*) = (0+1) / (8+6) = 1/14

P(Chinese|*j*) = (1+1) / (3+6) = 2/9

P(Tokyo|*j*) = (1+1) / (3+6) = 2/9

P(Japan|*j*) = (1+1) / (3+6) = 2/9

Choosing a class:

$P(c|d5) \propto 3/4 * (3/7)^3 * 1/14 * 1/14 \approx 0.0003$

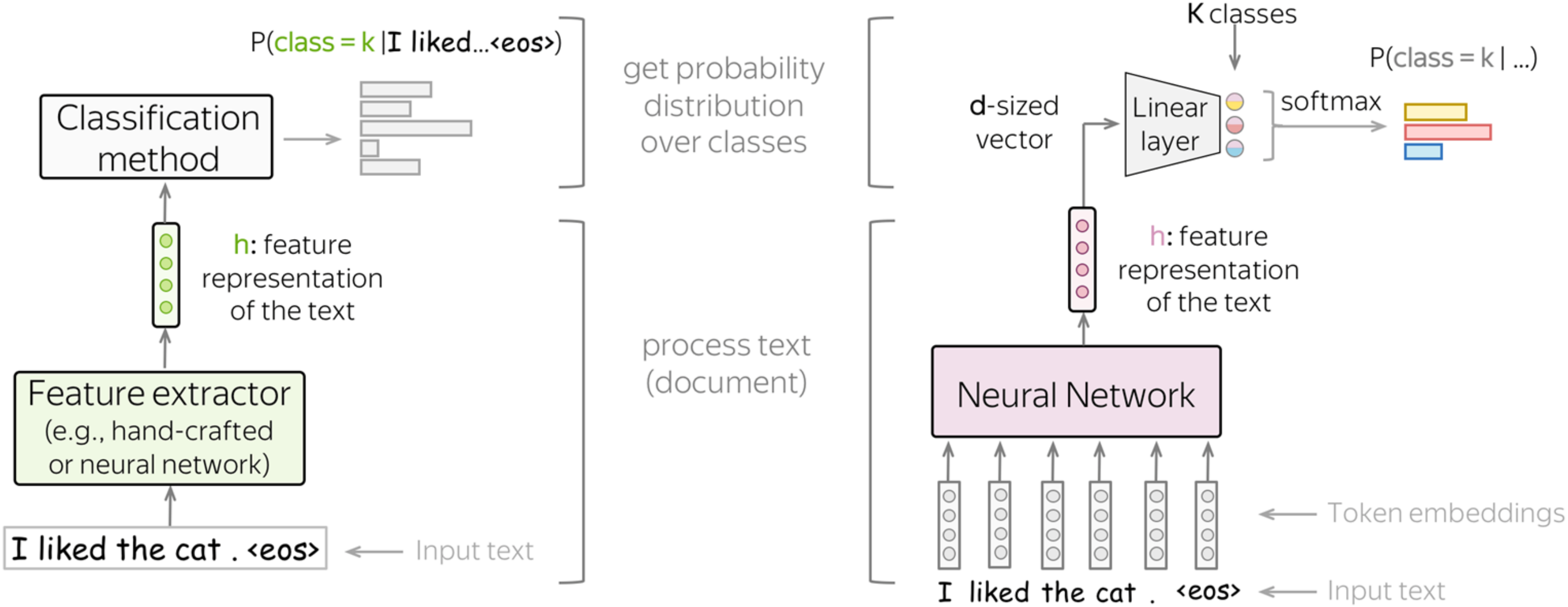
$P(j|d5) \propto 1/4 * (2/9)^3 * 2/9 * 2/9 \approx 0.0001$

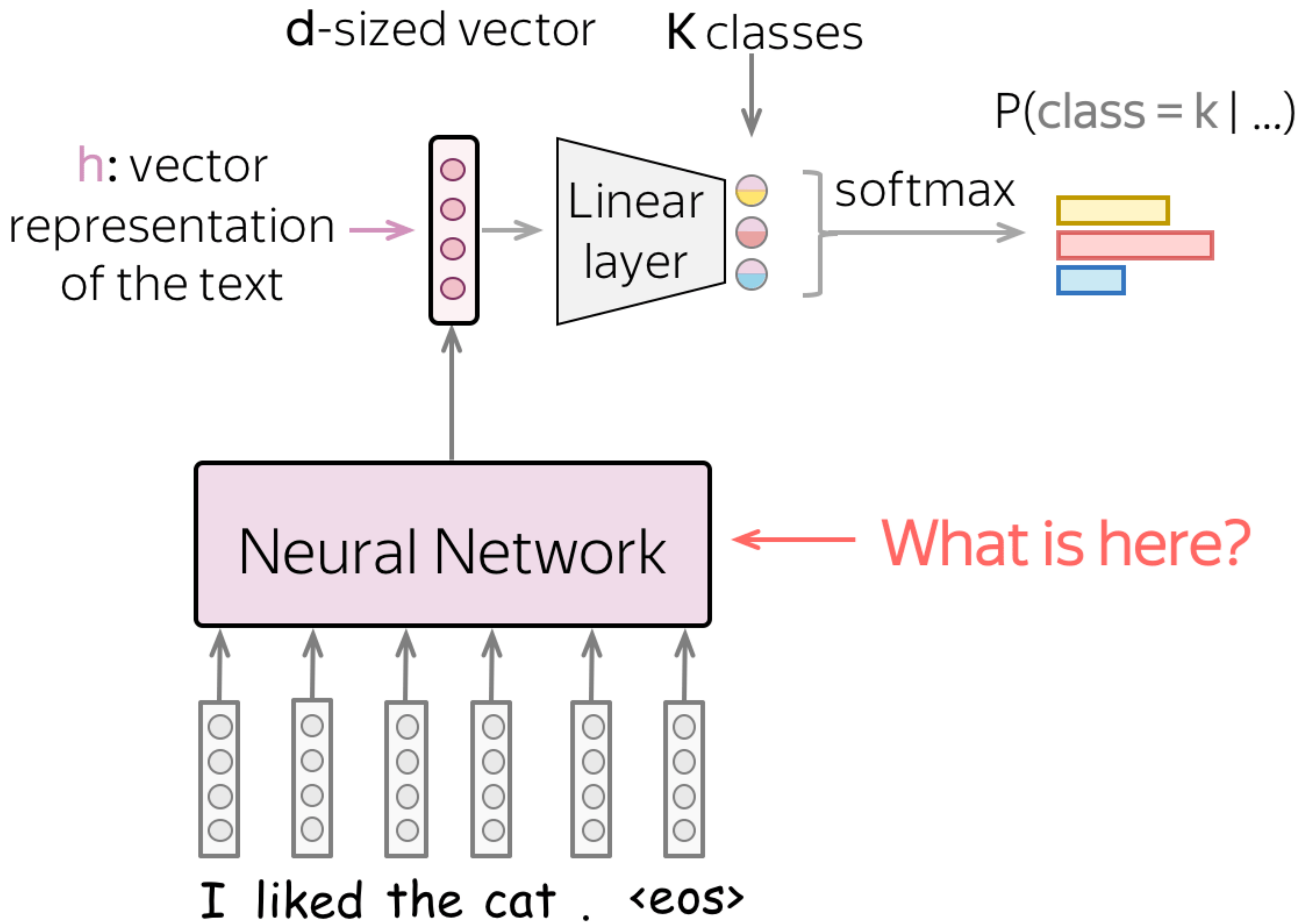
Text Classification with Neural Networks

- Feature representation of the input text can be obtained using a neural network
- We feed the embeddings of the input tokens to a neural network, and this neural network gives us a vector representation of the input text.
- After that, this vector is used for classification.

General Classification Pipeline

Classification with Neural Networks




d-sized vector
K classes
P(class = k | ...)
h: vector representation of the text
Linear layer
softmax
Neural Network
What is here?
I liked the cat . <eos>

Thank you