Text Classification NER, POS

Prashant K. Sharma

Diptesh Kanojia

https://shala2020.github.io/

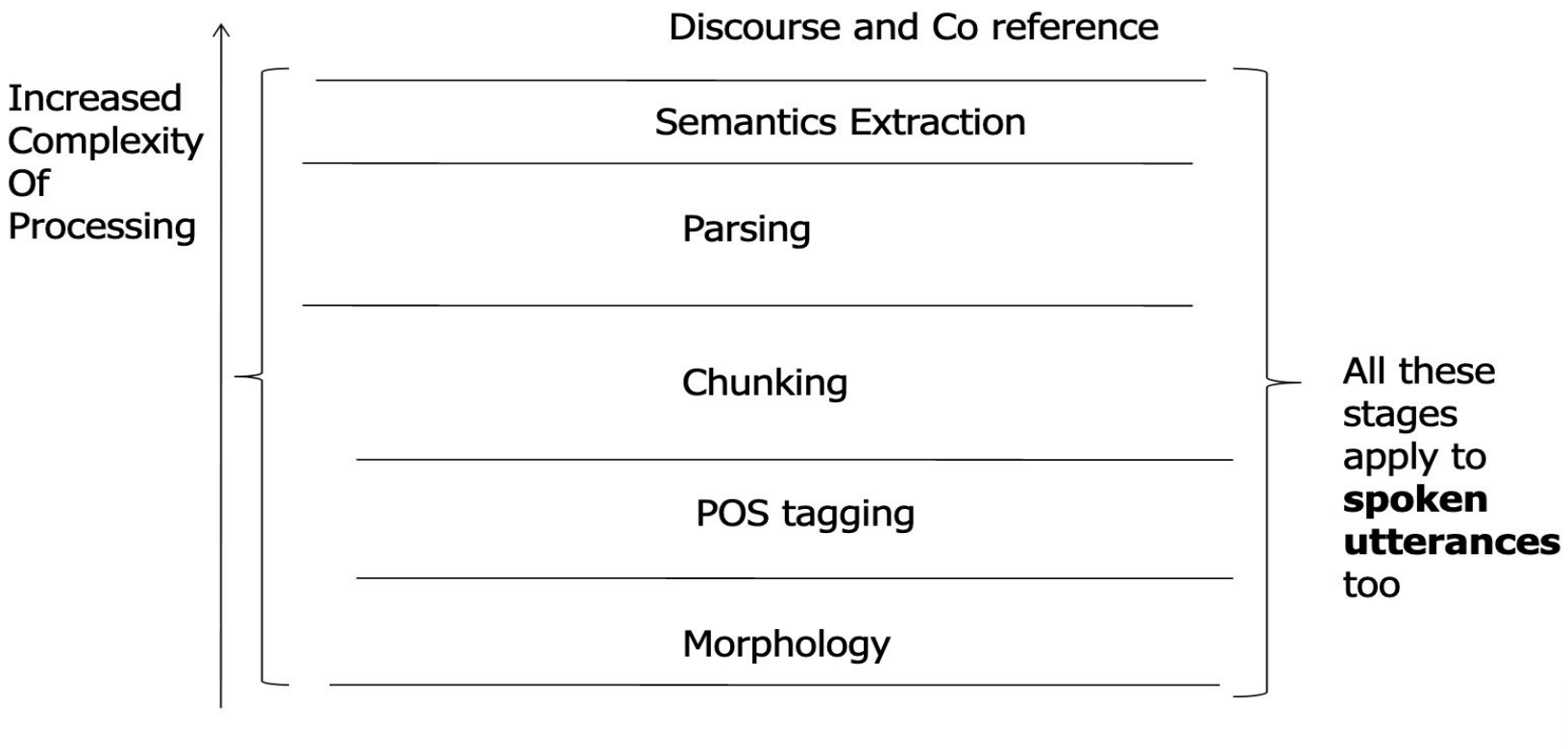
We will start at 09.05 pm

Agenda

- Text Classification
- Classical approaches for text representation
- Modern approaches for text representation
- Challenges in NLP : NER, POS

Introduction to Text Classification

NLP and Speech Layers



Credit: Prof. Pushpak Bhattacharyya

NLP and Speech Layers

- Morphology : karna / try - tried / visible - invisible / create - recreate

  An example

- POS : ram.NN kicked.?? the.?? ball.??
- Chunking : the book has many chapters / noun compounds
- Parsing
- Semantics Extraction - Word Senses / Word Vectors
- Discourse - Coreference Resolution / Sarcasm Detection

Credit: Prof. Pushpak Bhattacharyya

Why do we need to classify Texts ?

- Spam Classification

Why do we need to classify Texts ?

- Sentiment Analysis

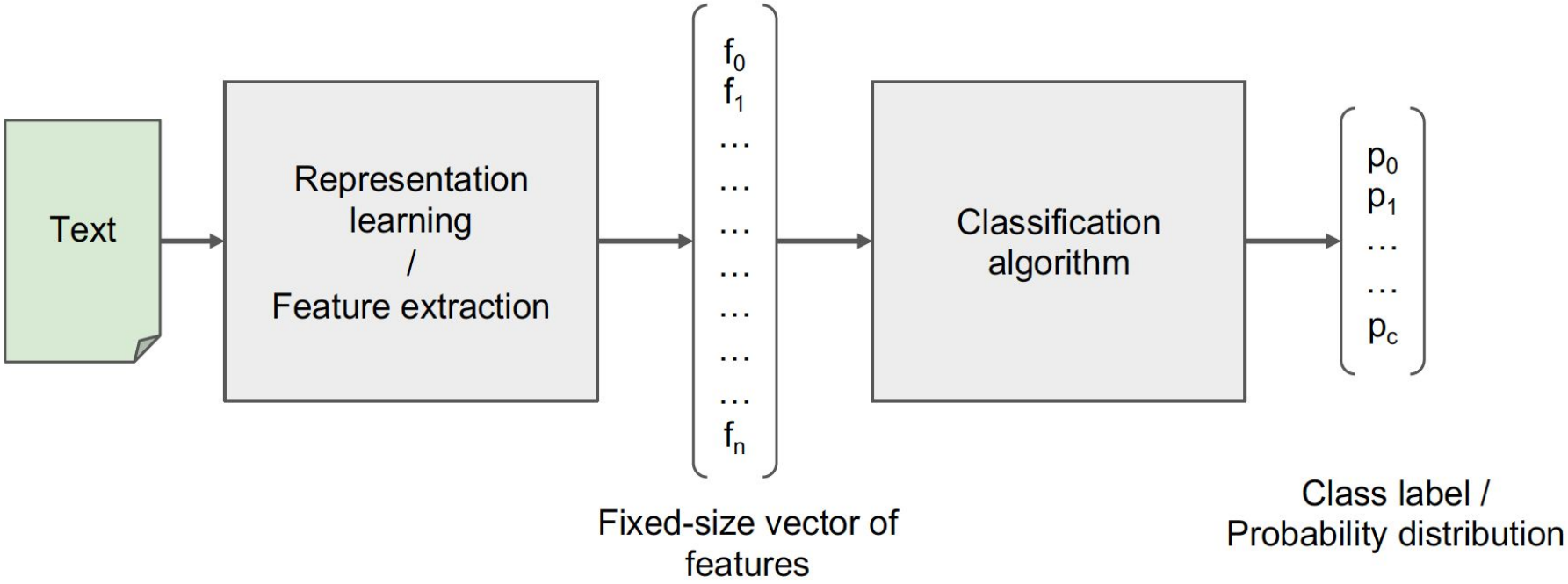
Why do we need to classify Texts ?

- Fake News/Click bait Detection

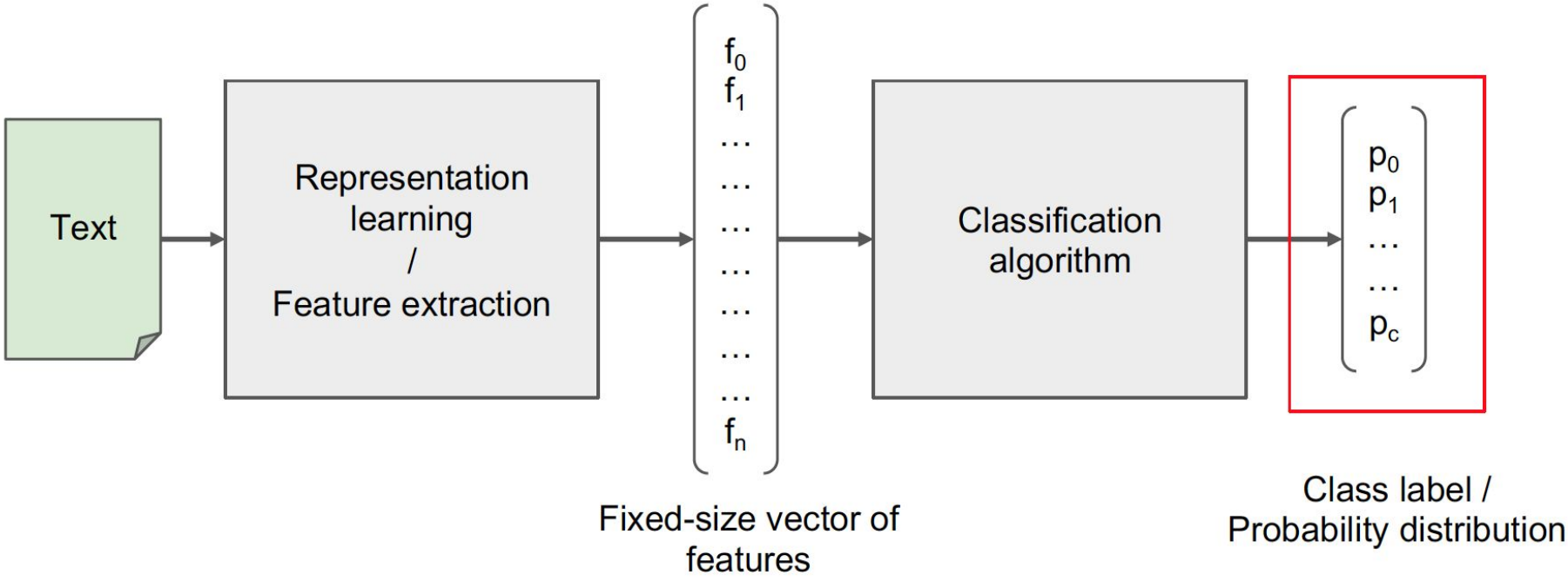
Why do we need to classify Texts ?

- Data Filtering

Text Classification in general



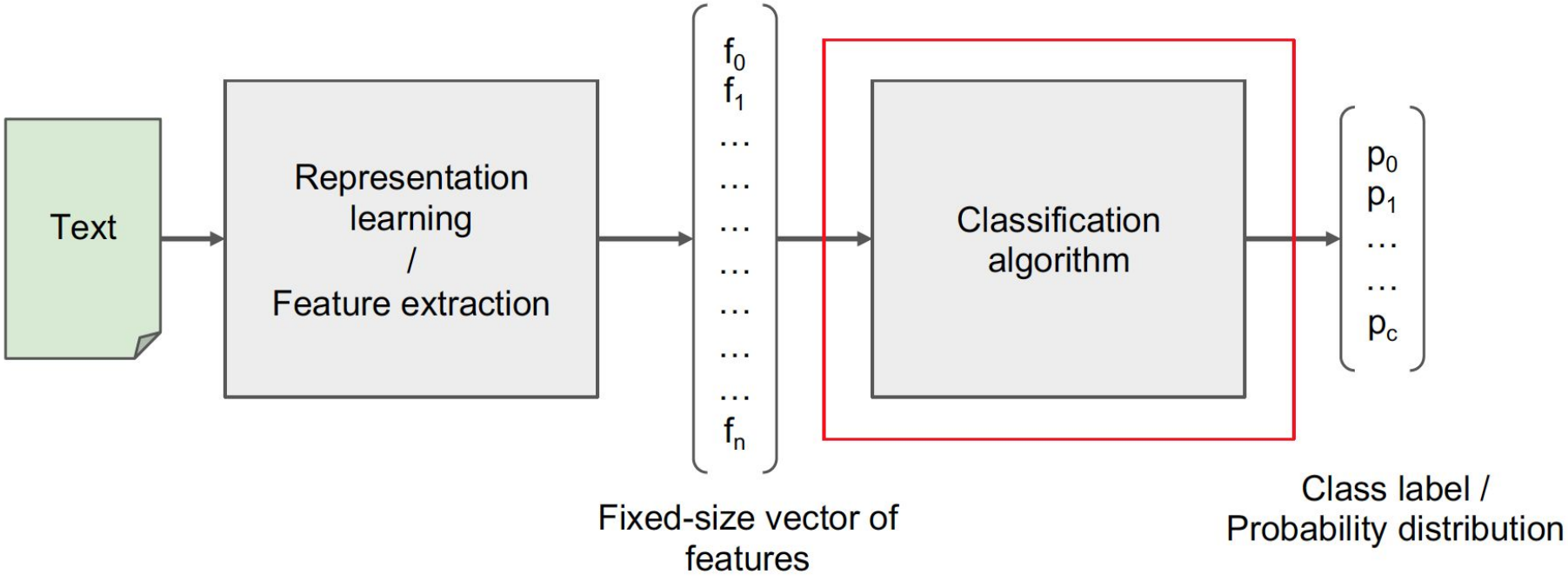
Text Classification in general



Text Label types

1. Discrete labels:
    a. Label is known:
        i. Binary classification: spam filtering/sentiment analysis
        ii. Multi-class classification: categorization of goods
        iii. Multi-label classification: #hashtag prediction
    b. Label is unknown:
        i. Text clasterization: user intent search
2. Continuous labels: predict a salary by CV, predict a price by a product description

Text Classification in general



Classification Algorithms

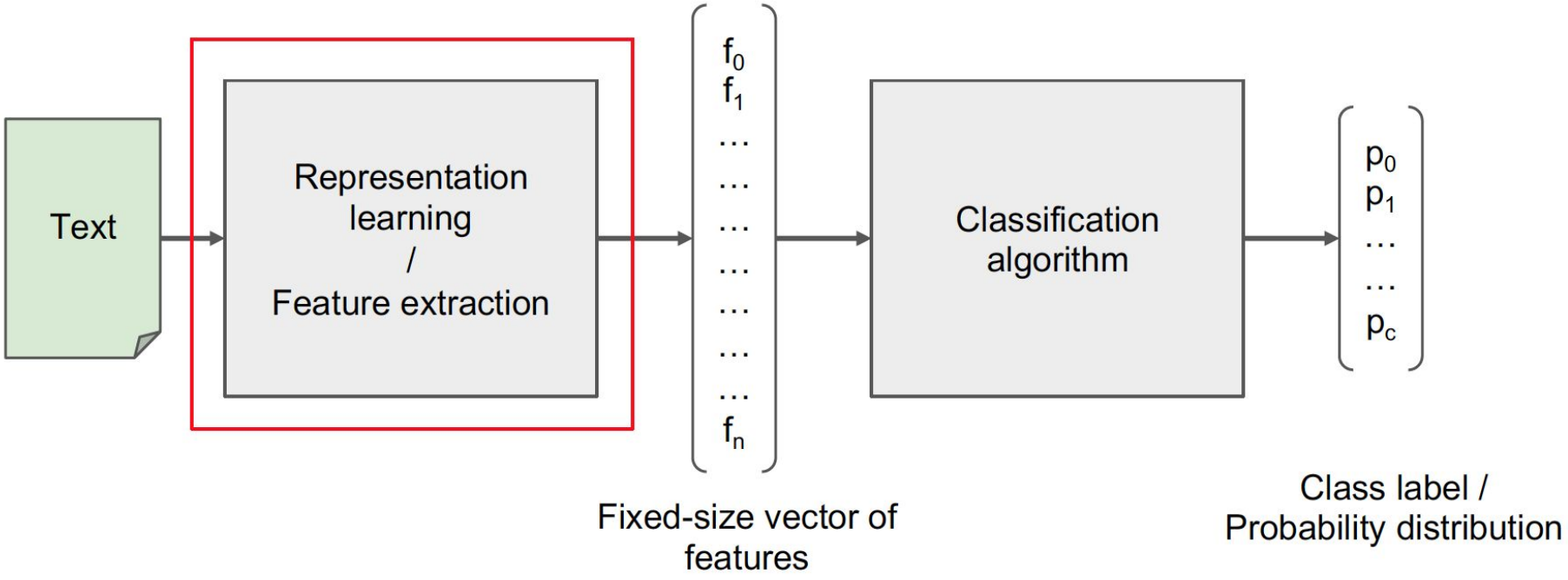
NLP has its own cozy atmosphere when it comes to naming models

TL;DR

Maximum Entropy Method = sklearn.linear_model.LogisticRegression

Multinomial Naive Bayes = sklearn.naive_bayes.MultinomialNB

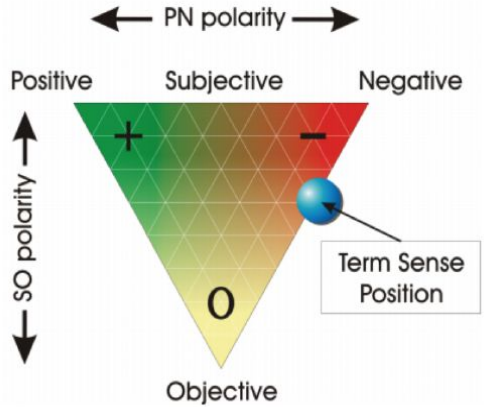
Text Classification in general



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 )



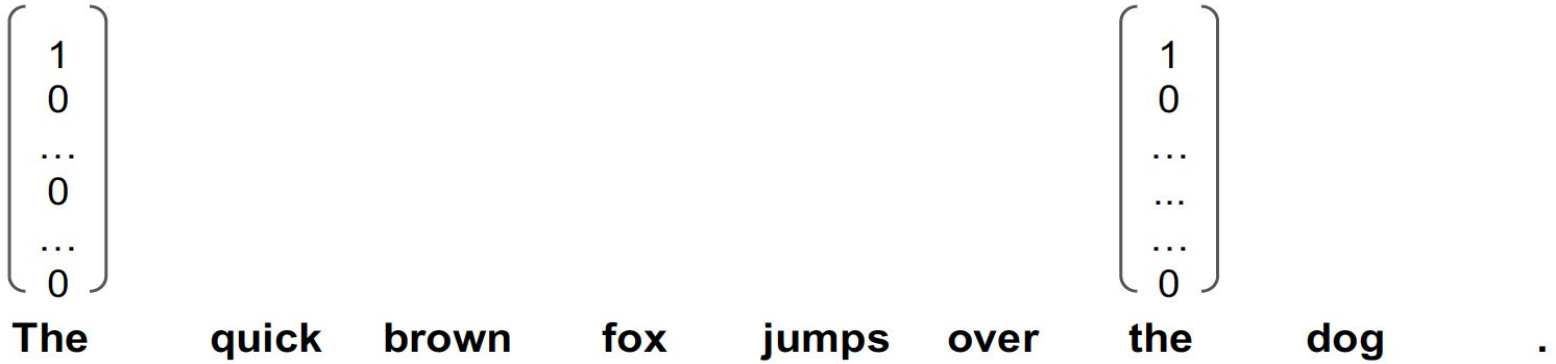
Sparse Text Representation: BOW

The	quick	brown	fox	jumps	over	~~lazy~~ the	dog	.

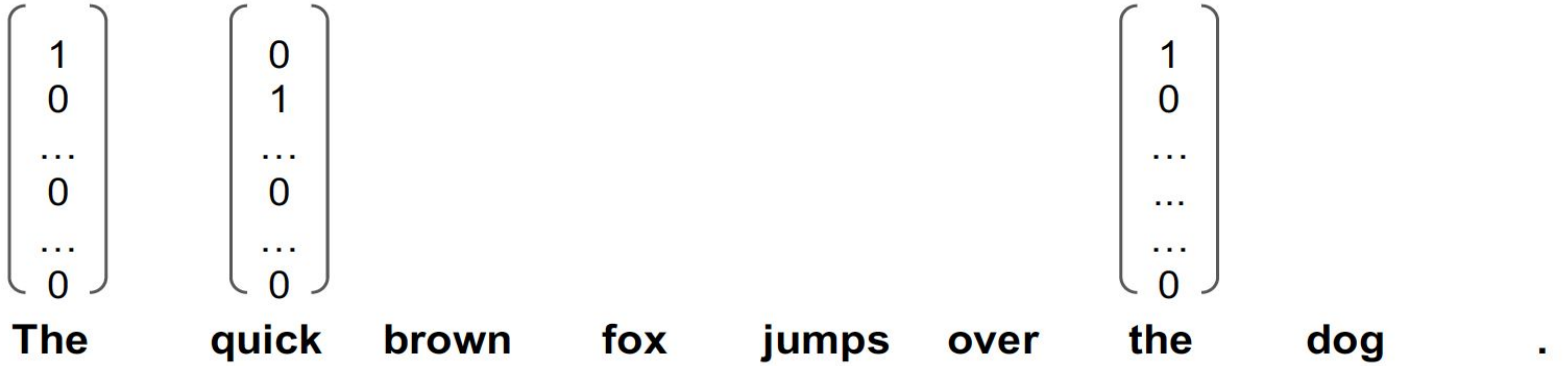
Sparse Text Representation: BOW

The quick brown fox jumps over the dog .

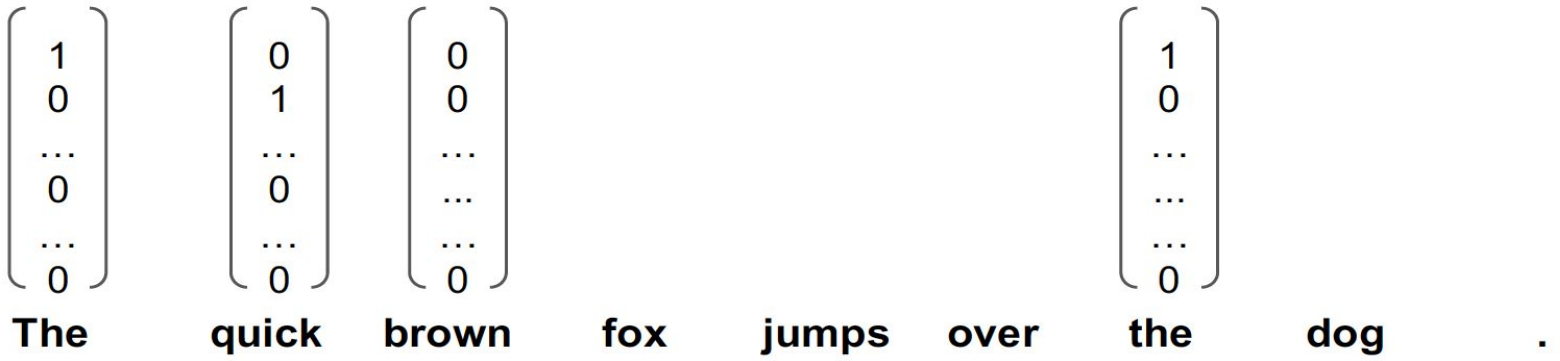
Sparse Text Representation: BOW



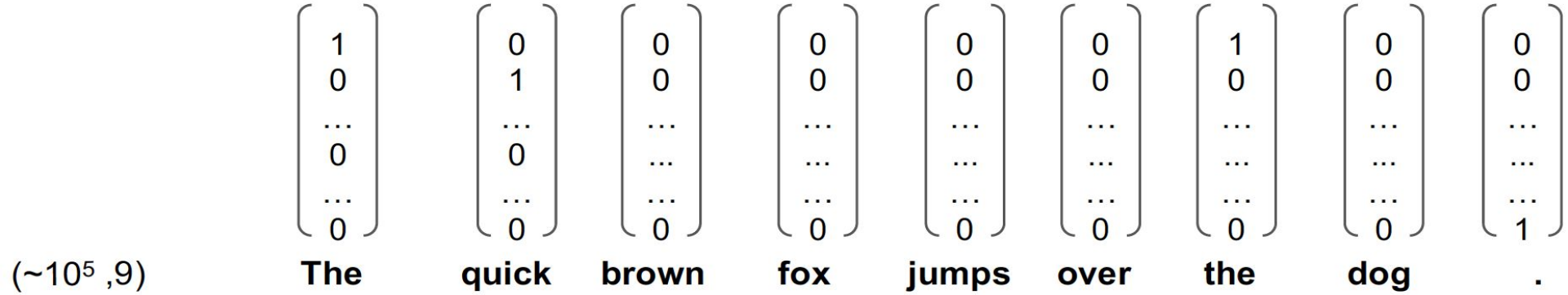
Sparse Text Representation: BOW



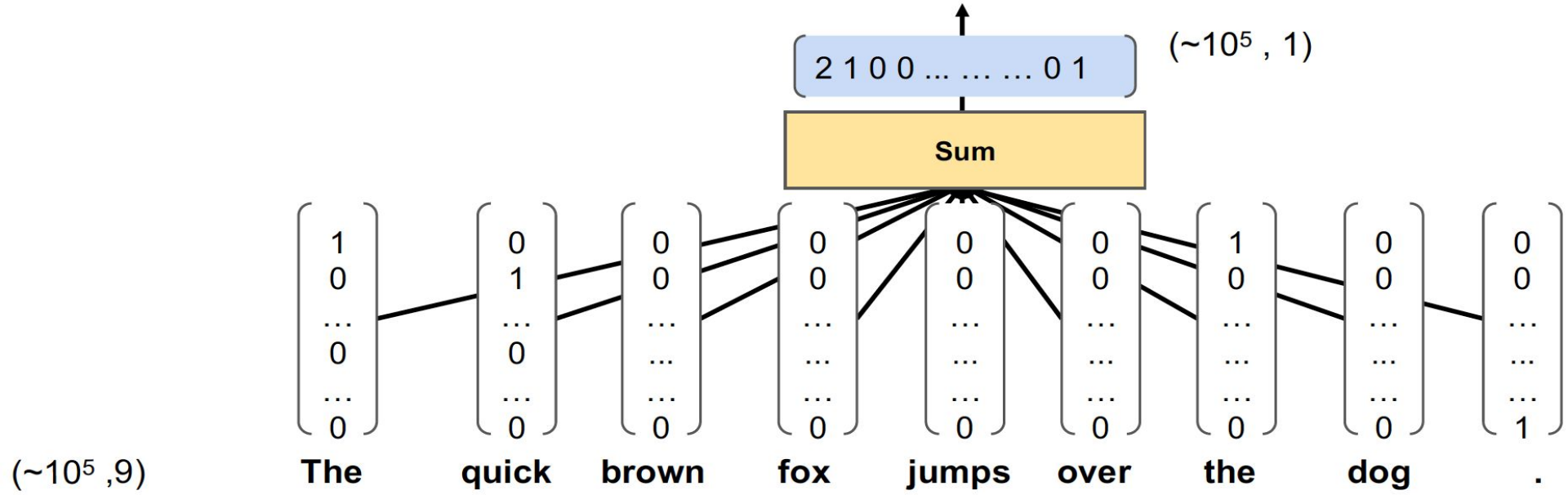
Sparse Text Representation: BOW



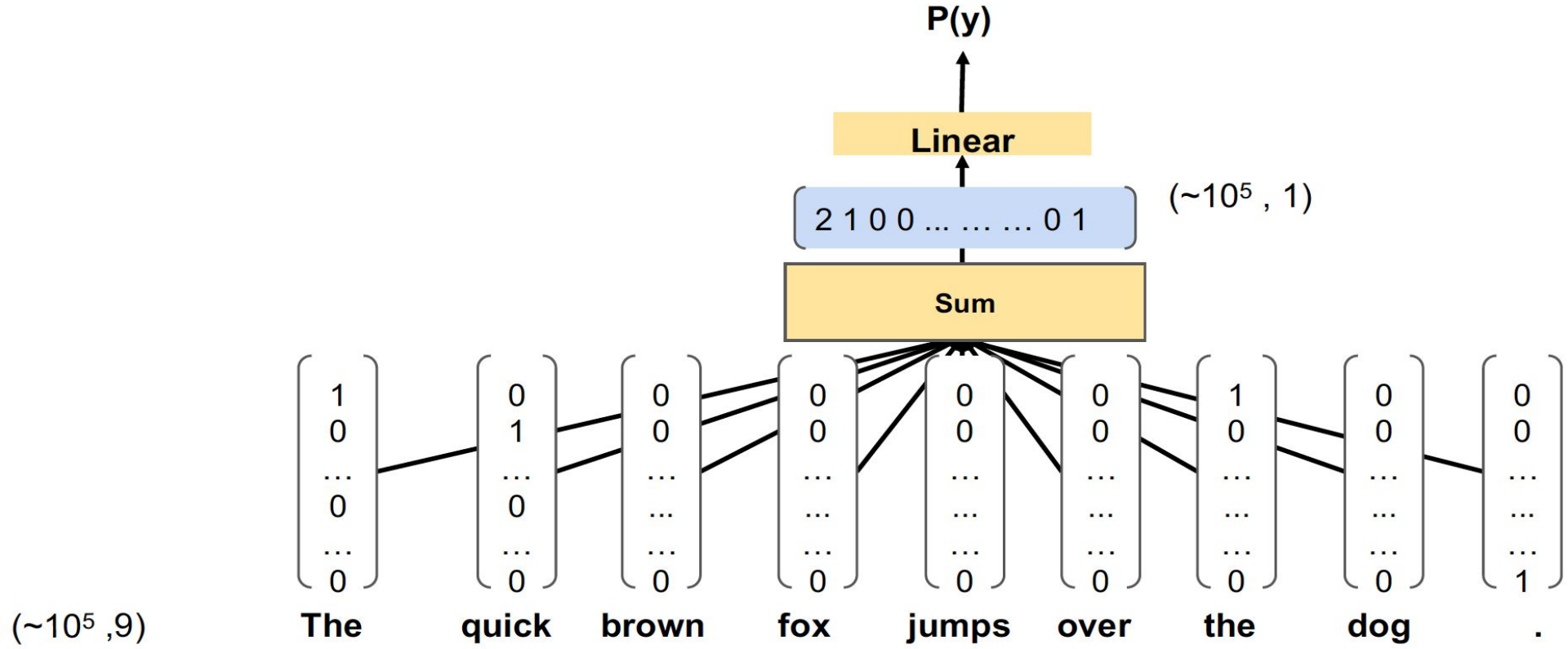
Sparse Text Representation: BOW



Sparse Text Representation: BOW



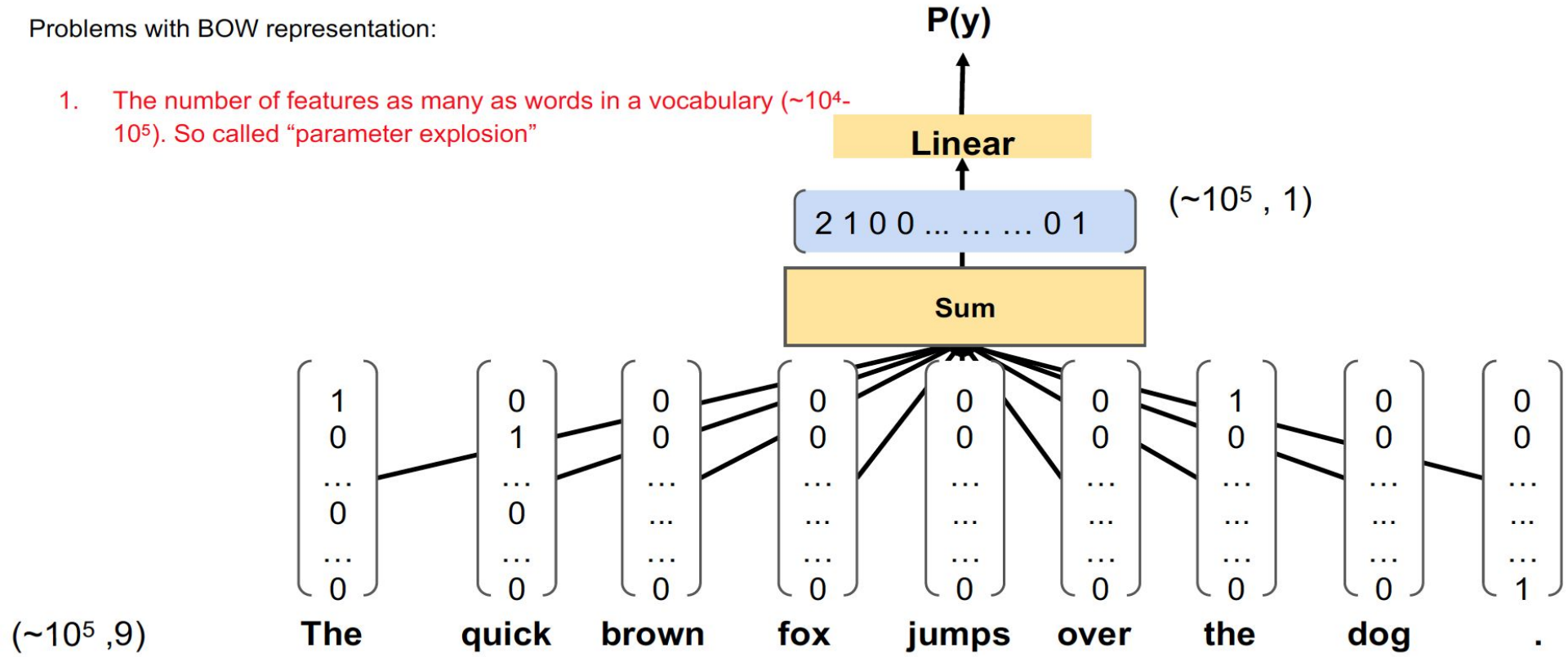
Sparse Text Representation: BOW



Sparse Text Representation: BOW

Problems with BOW representation:

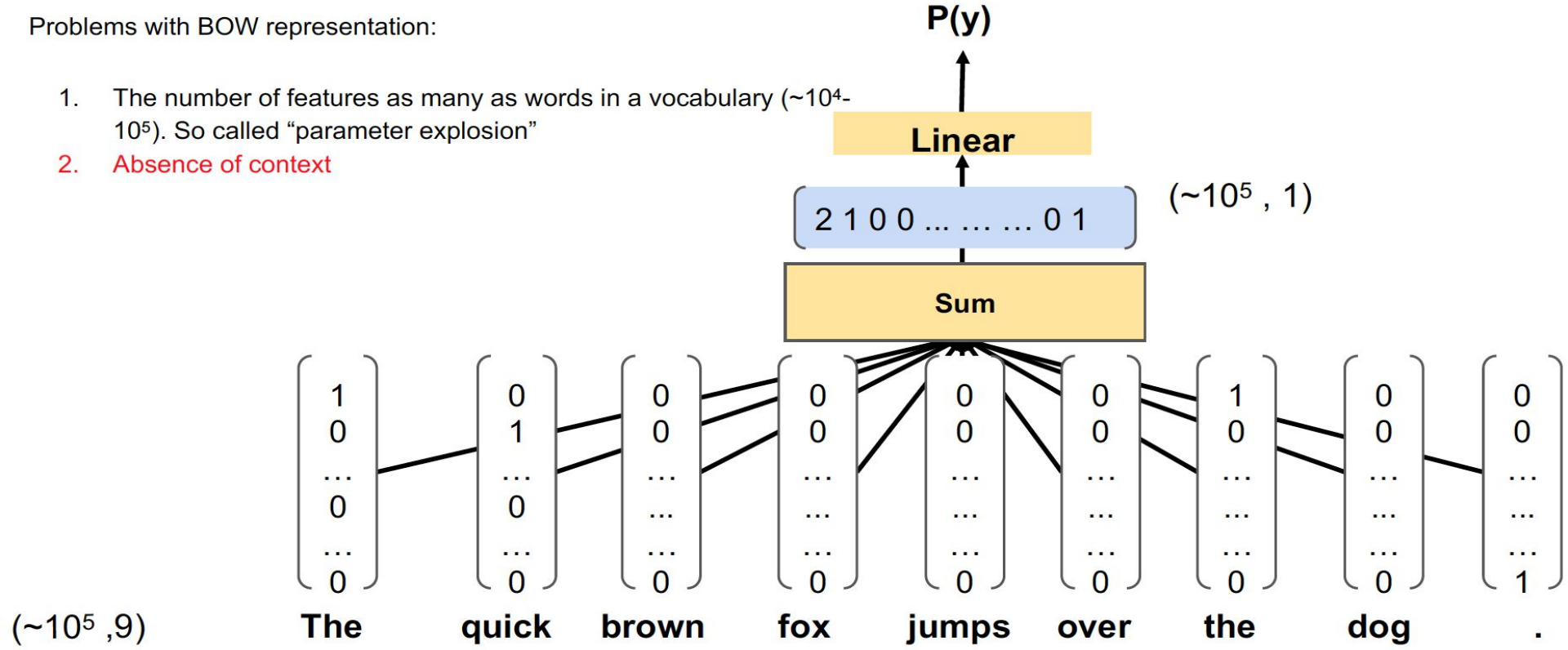
1. The number of features as many as words in a vocabulary (~10^4-10^5). So called “parameter explosion”



Sparse Text Representation: BOW

Problems with BOW representation:

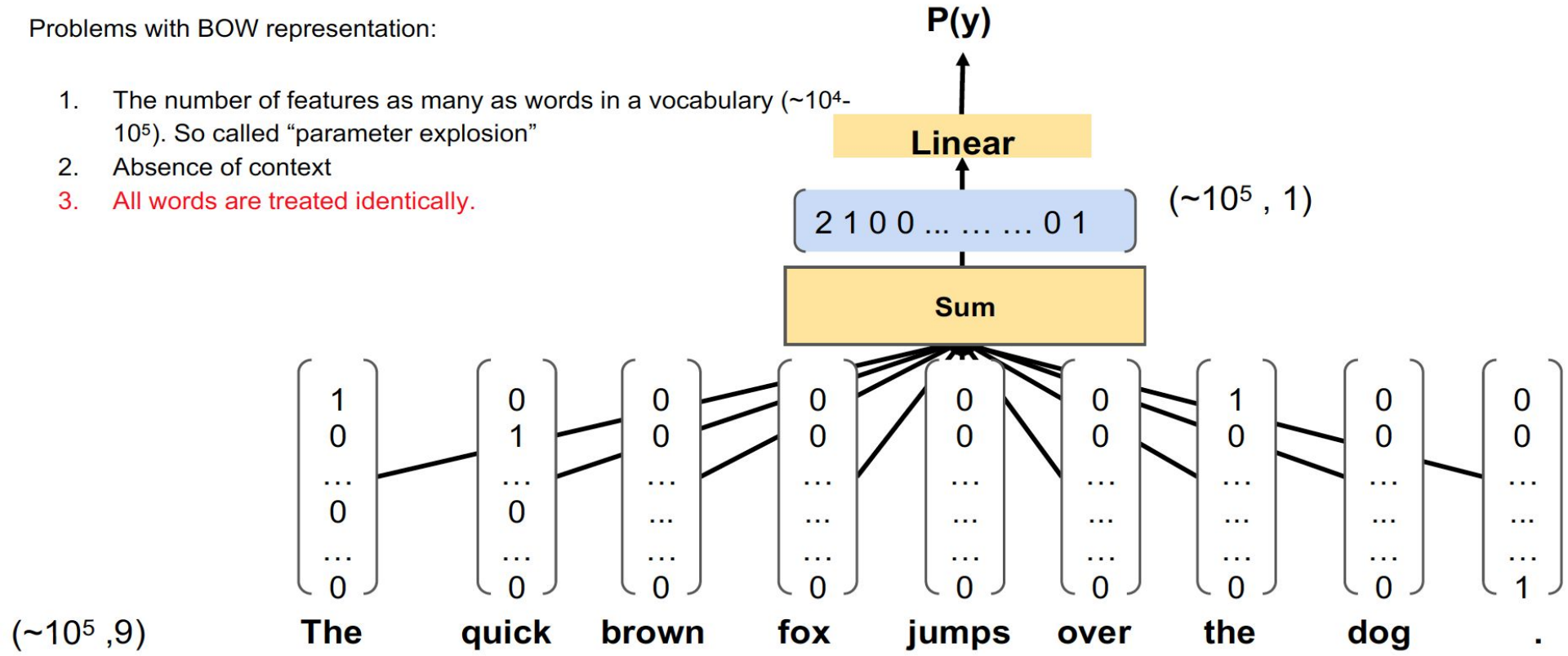
1. The number of features as many as words in a vocabulary (~10^4-10^5). So called “parameter explosion”
2. Absence of context



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.

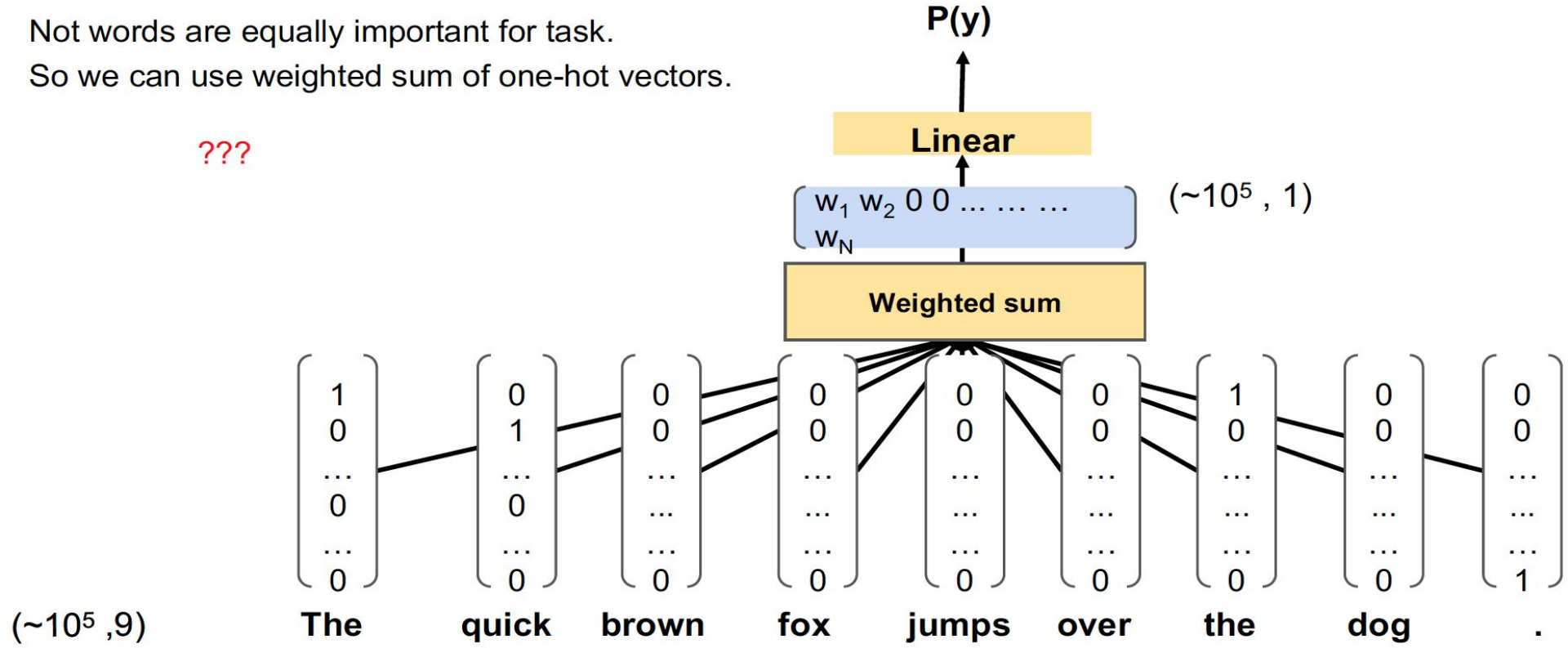


Weighting techniques for BOW

Not words are equally important for task.

So we can use weighted sum of one-hot vectors.

???

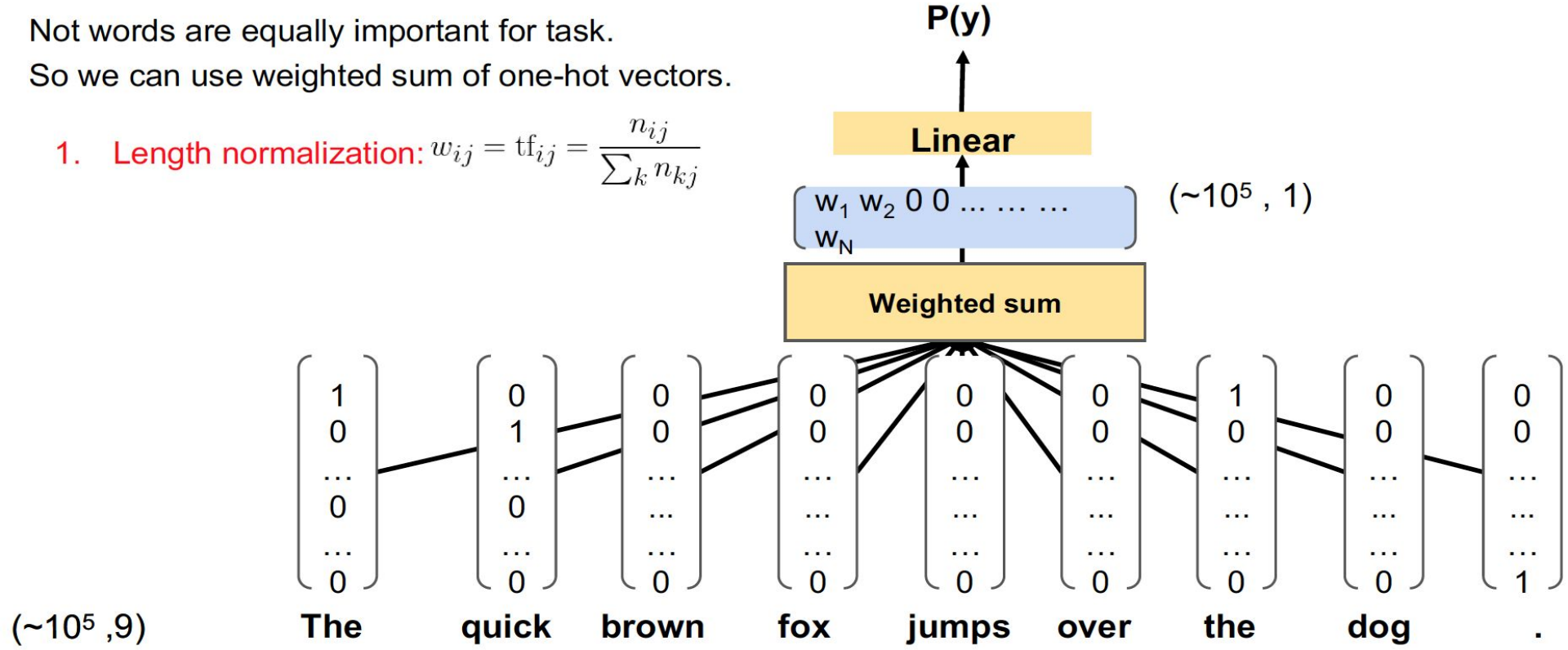


Weighting techniques for BOW

Not words are equally important for task.

So we can use weighted sum of one-hot vectors.

1. Length normalization: $w_{ij} = \mathrm{tf}_{ij} = \frac{n_{ij}}{\sum_k n_{kj}}$

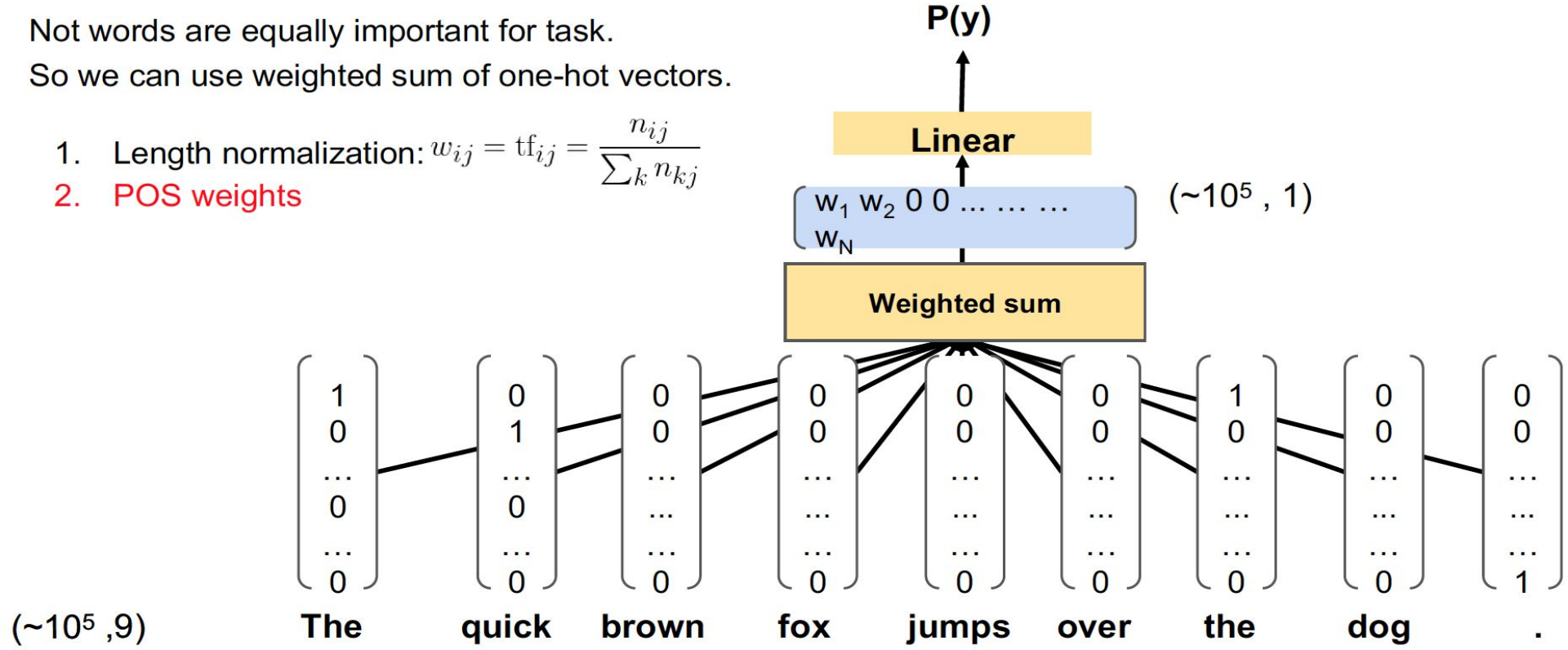


Weighting techniques for BOW

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

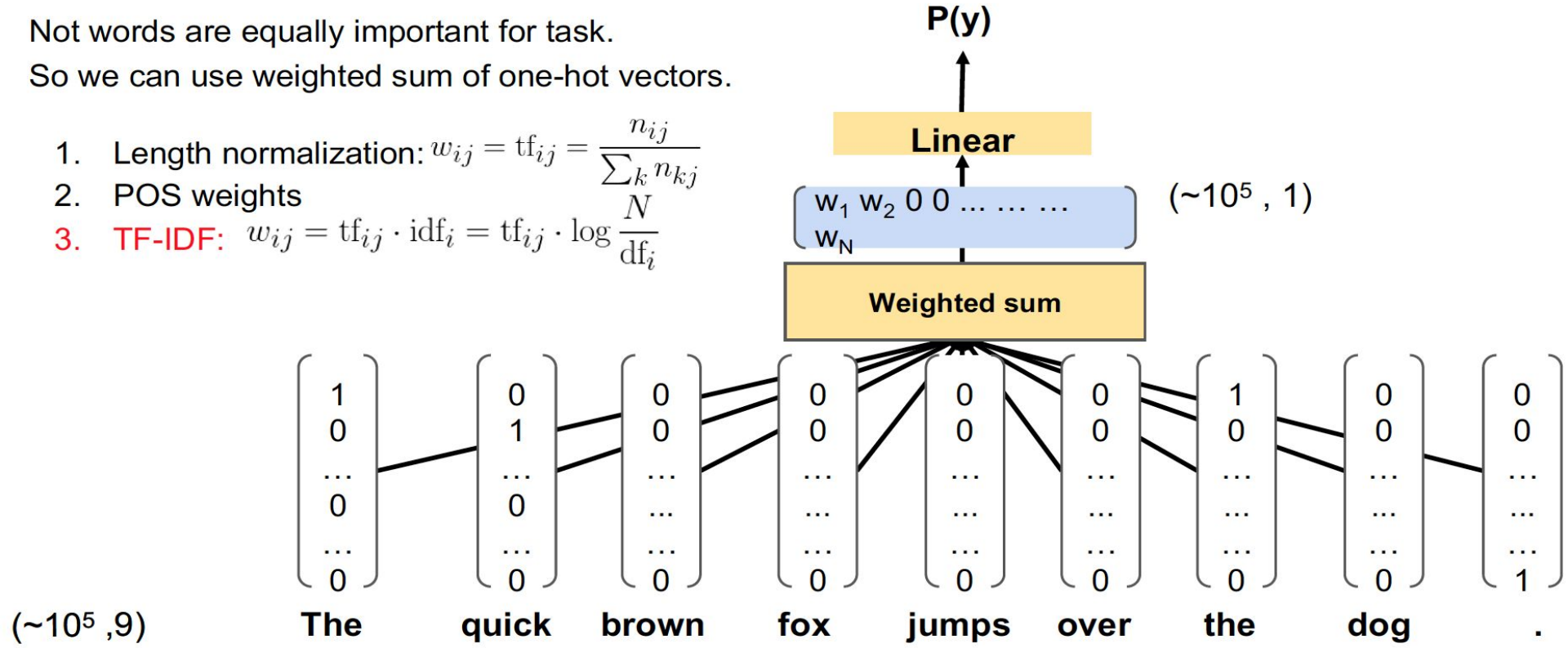


Weighting techniques for BOW

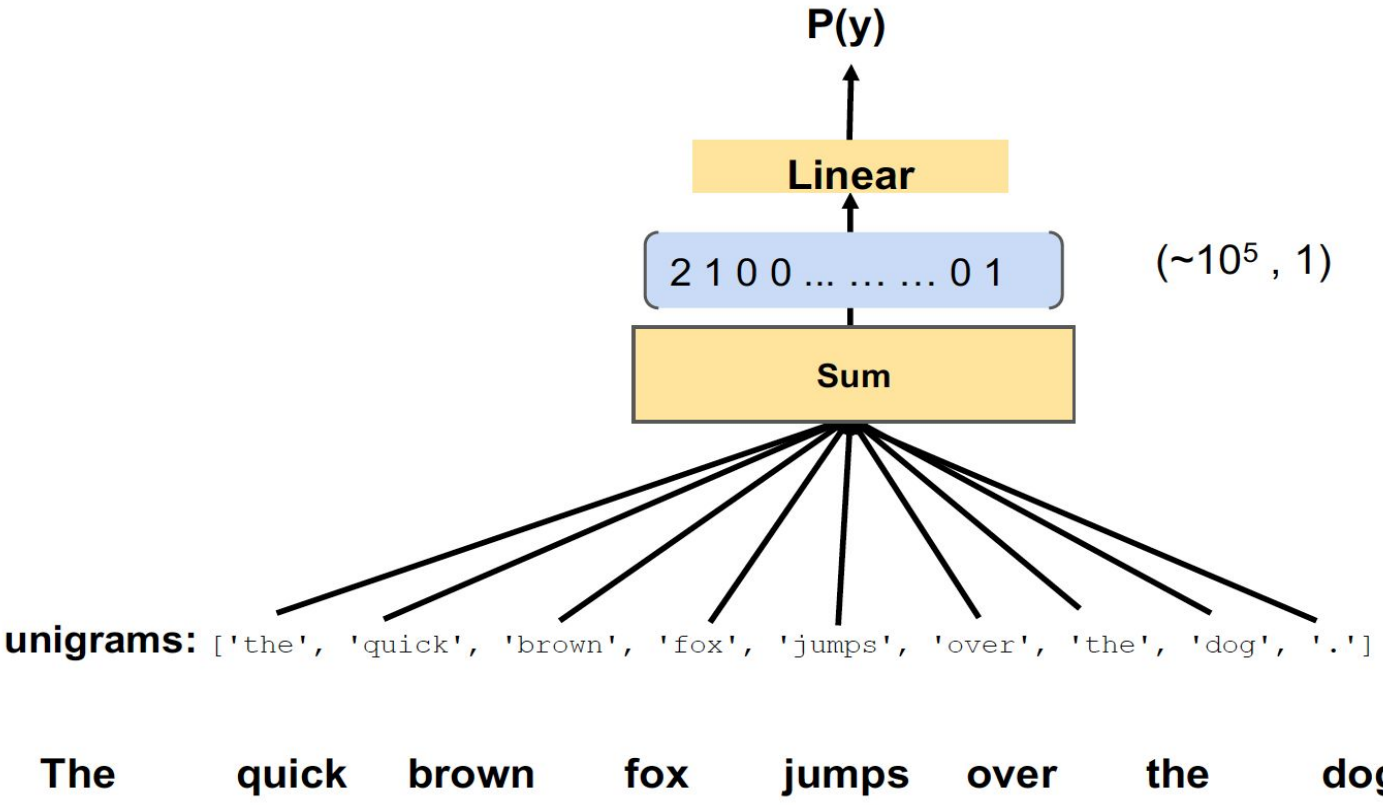
Not words are equally important for task.

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: $w_{ij} = \mathrm{tf}_{ij} \cdot \mathrm{idf}_i = \mathrm{tf}_{ij} \cdot \log \frac{N}{\mathrm{df}_i}$

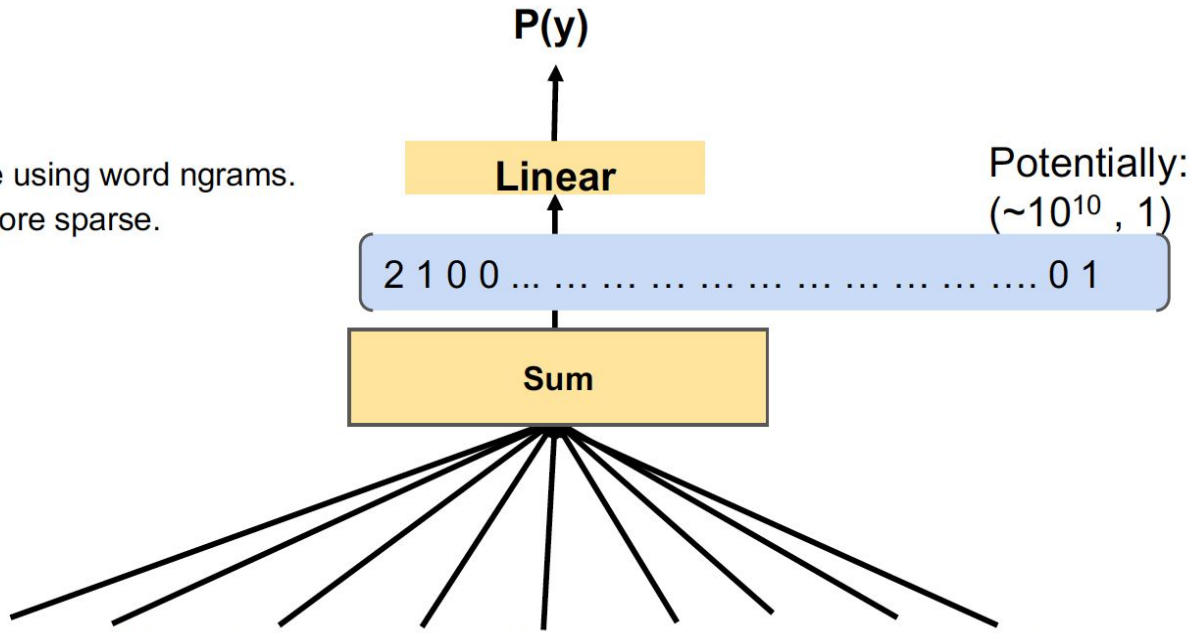


Context Importance



Context Importance

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



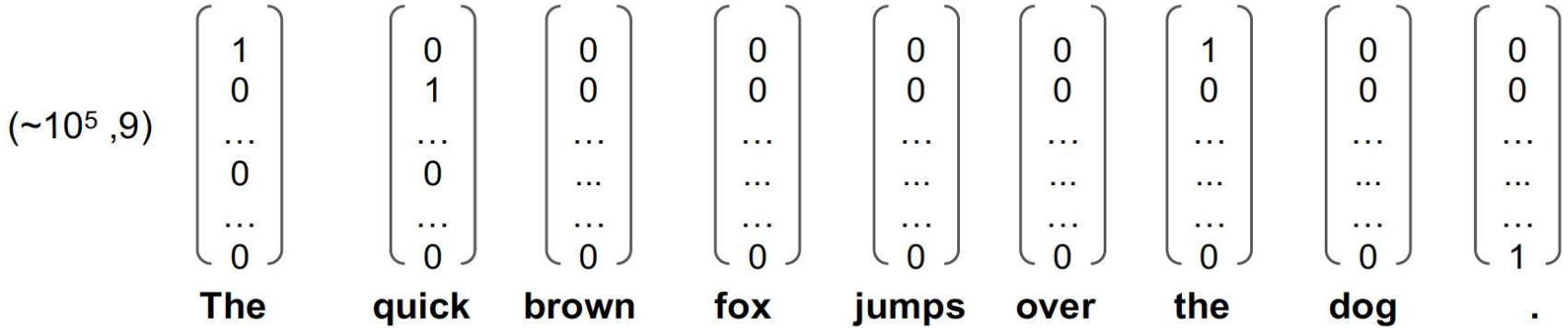
unigrams: `['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'dog', '.']`

bigrams: `['the quick', 'quick brown', 'brown fox', 'fox jumps', 'jumps over', 'over the', 'the dog', 'dog .']`

The quick brown fox jumps over the dog .

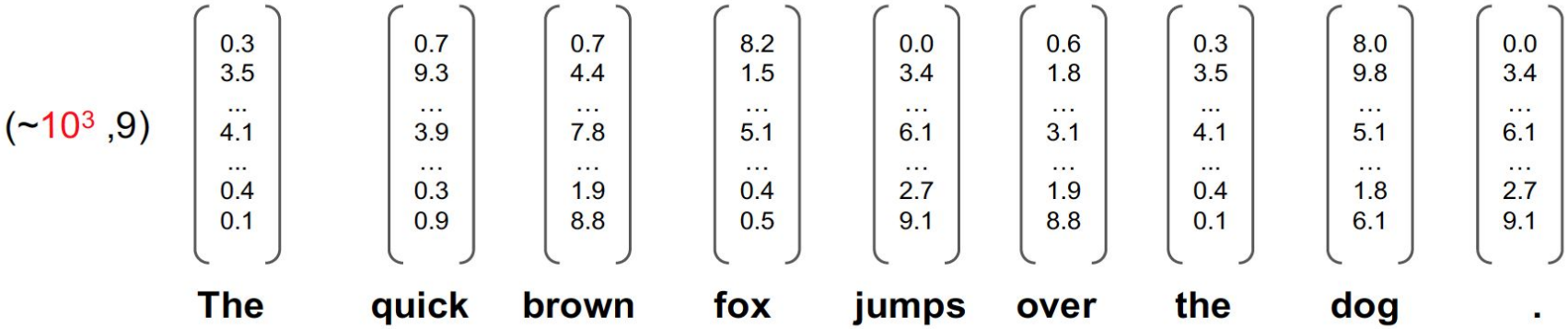
Dense Text Representation: NBOW

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



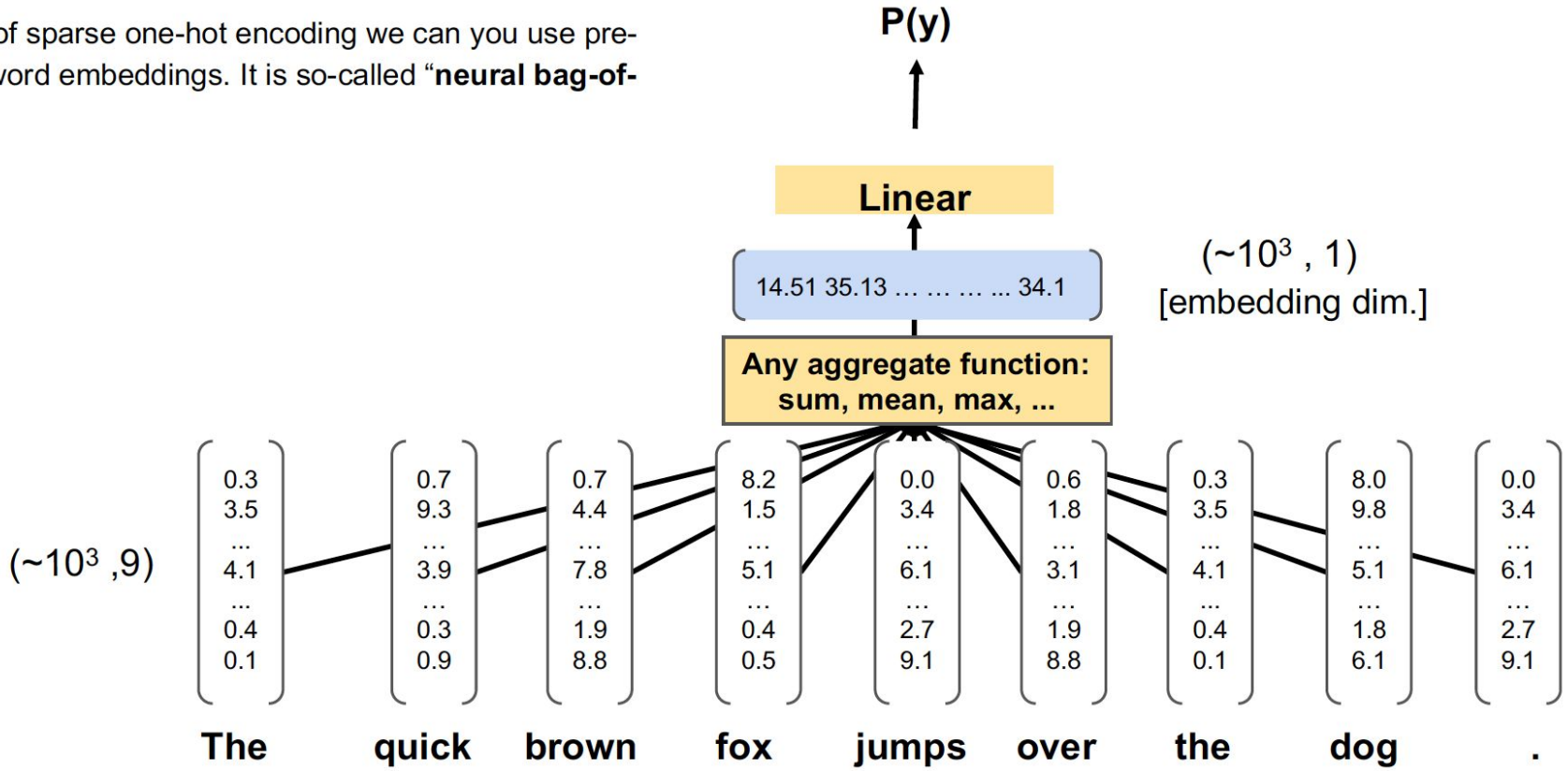
Dense Text Representation: NBOW

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



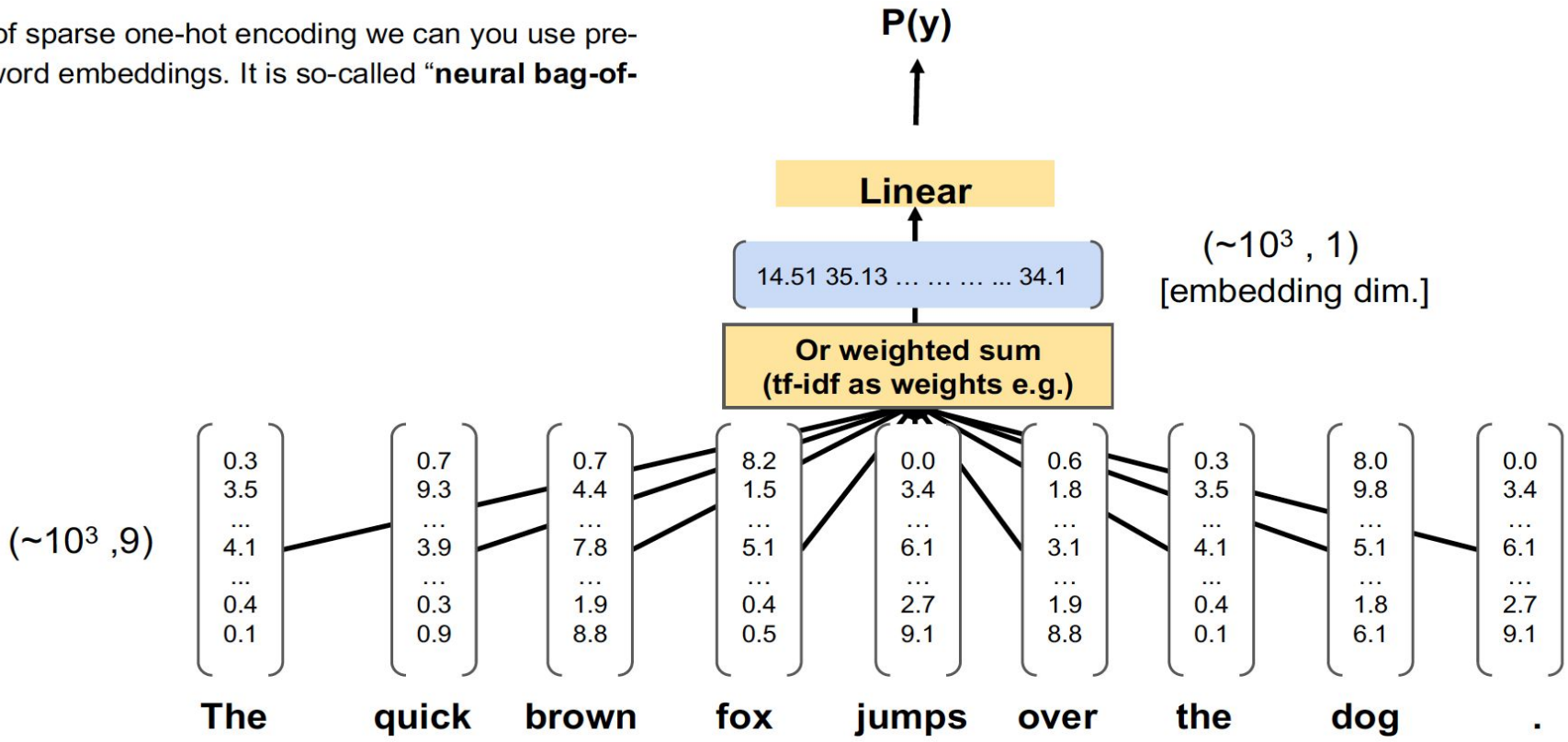
Dense Text Representation: NBOW

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



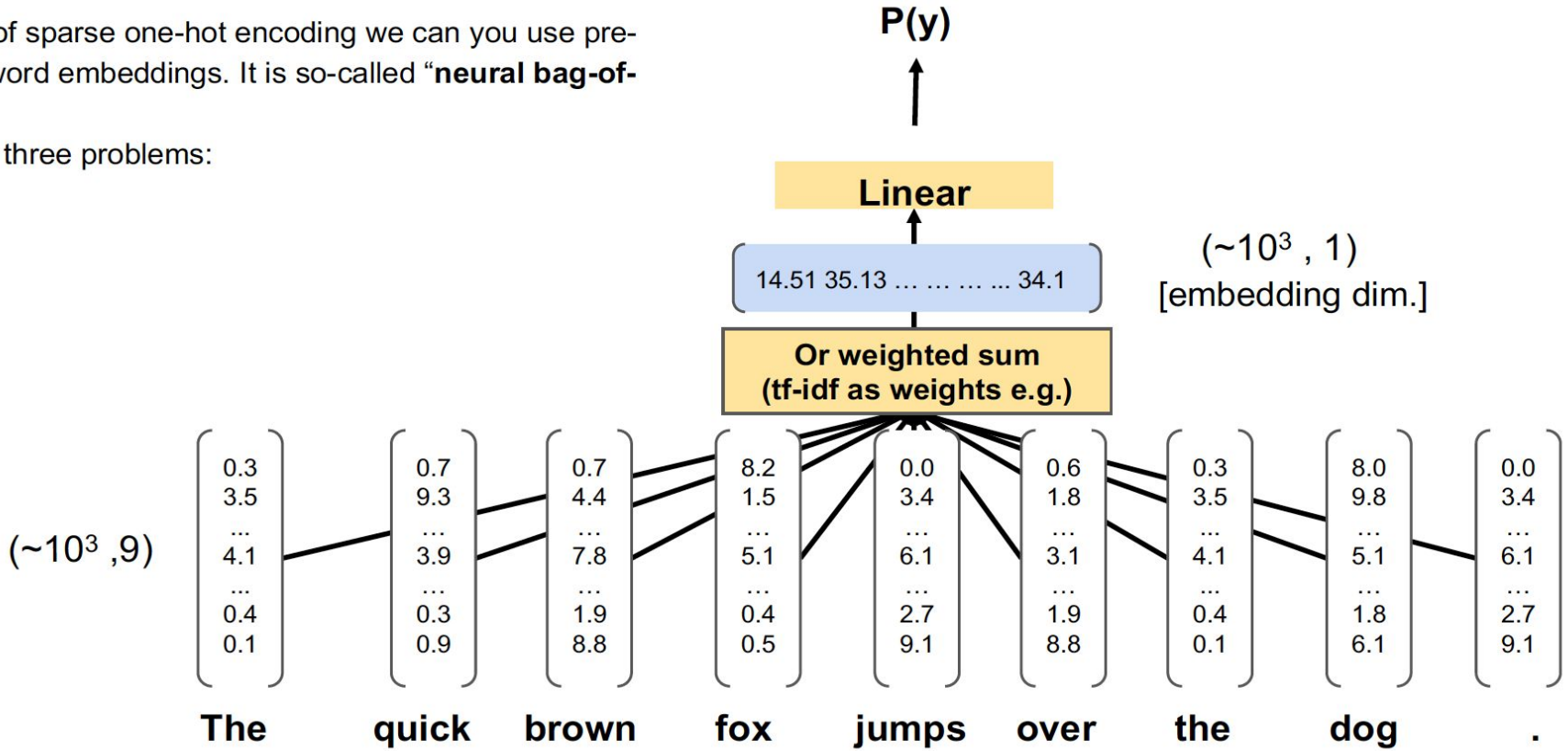
Dense Text Representation: NBOW

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

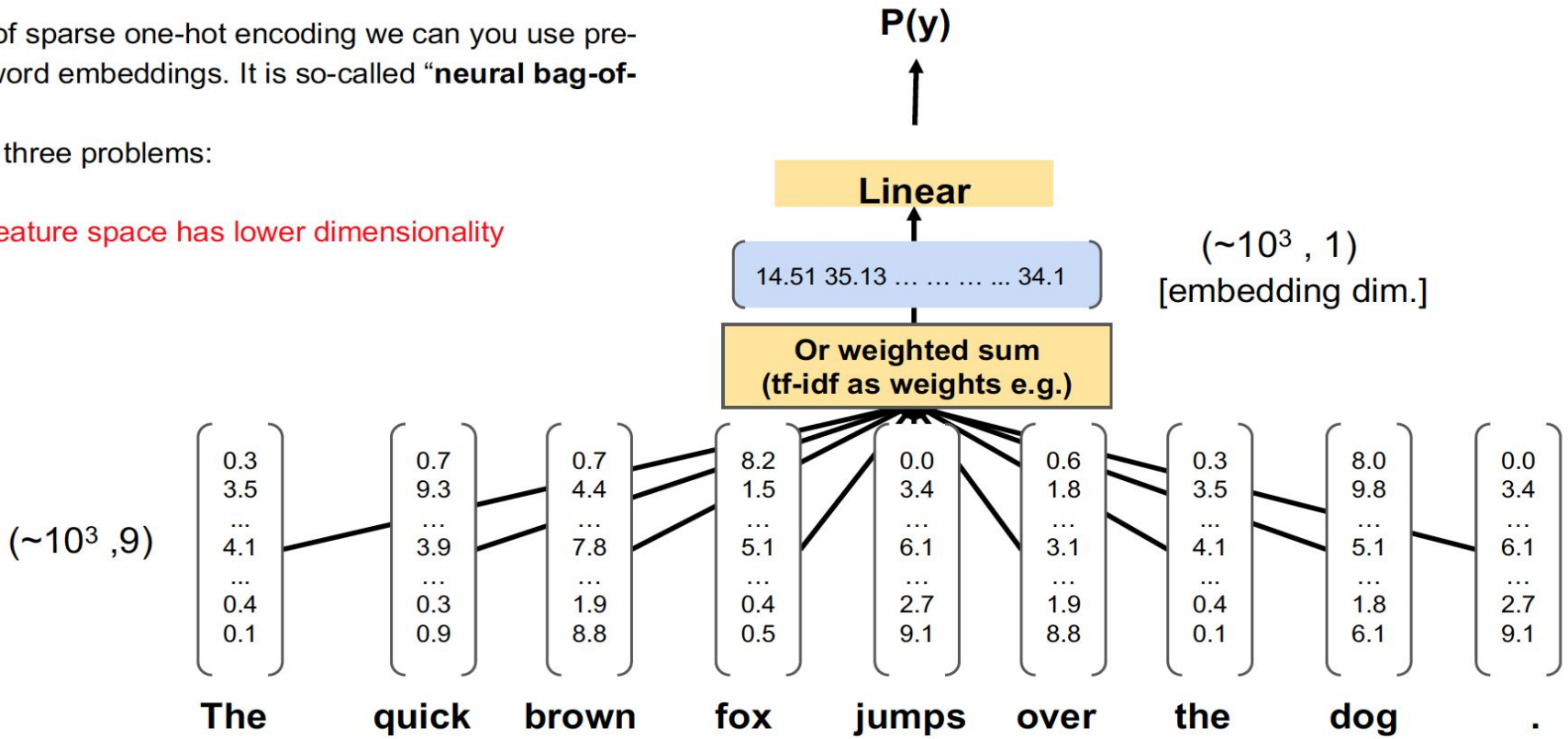
Instead of sparse one-hot encoding we can you use pre-trained word embeddings. It is so-called "**neural bag-of-words**".
It solves three problems:



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:

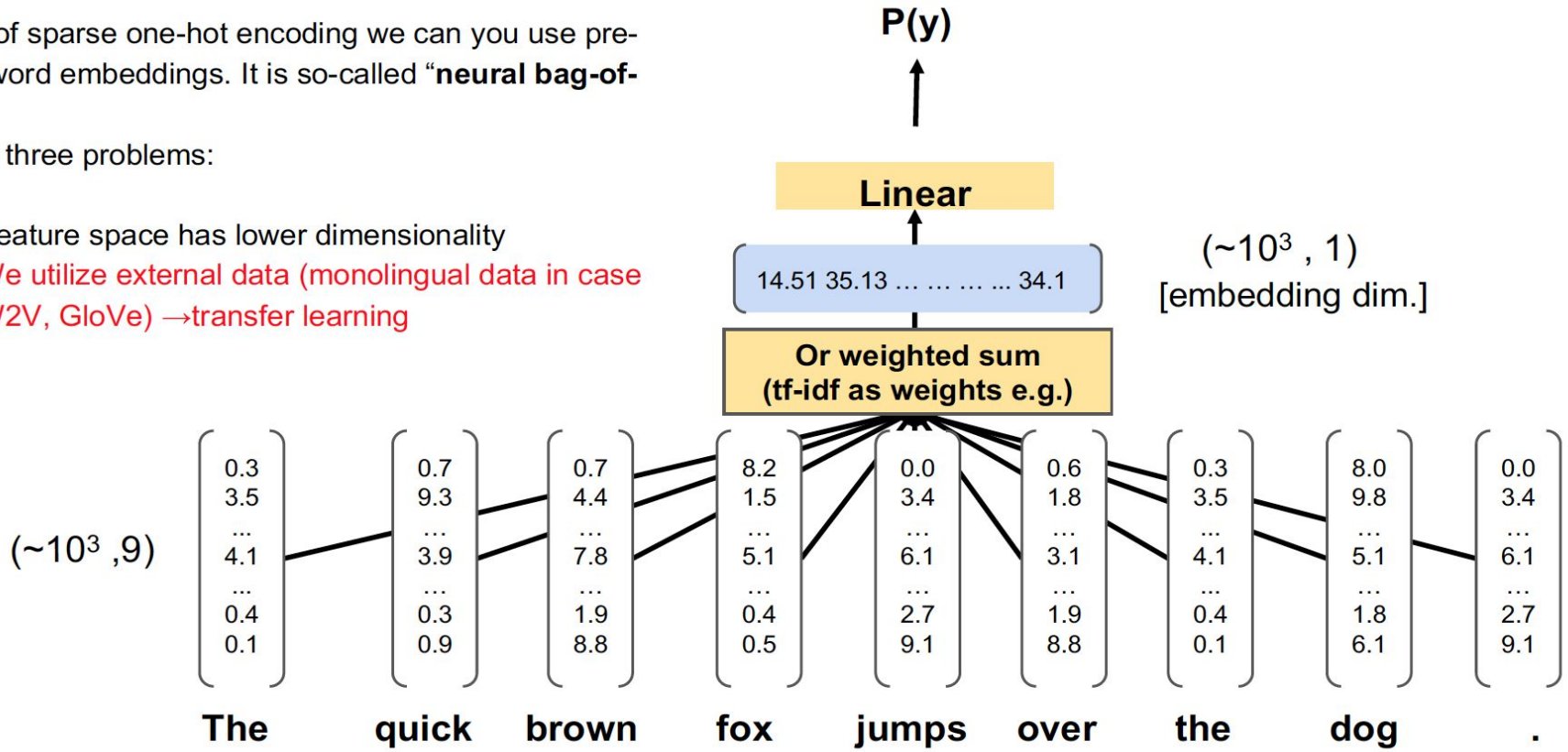
1. Feature space has lower dimensionality



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:

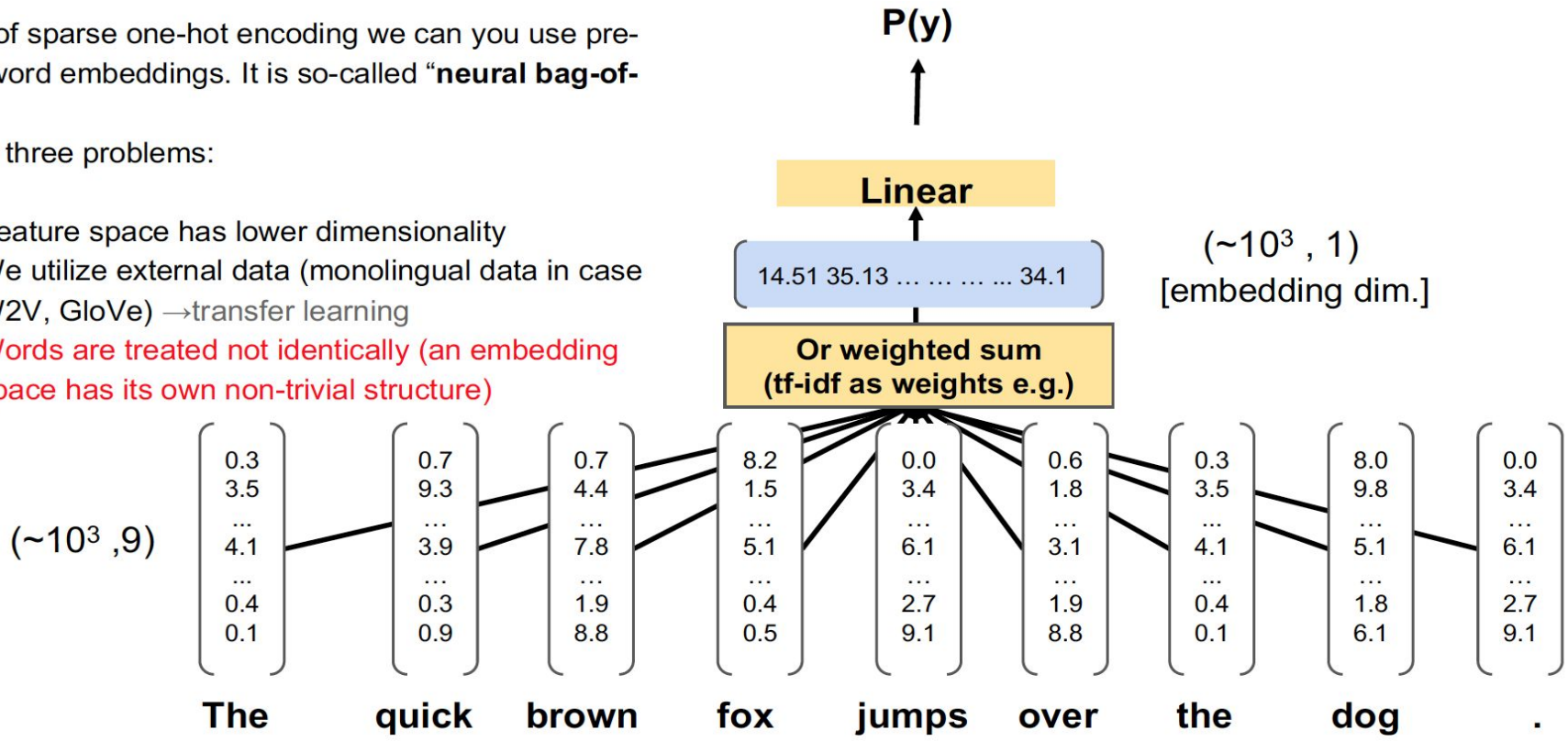
1. Feature space has lower dimensionality
2. We utilize external data (monolingual data in case W2V, GloVe) →transfer learning



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)

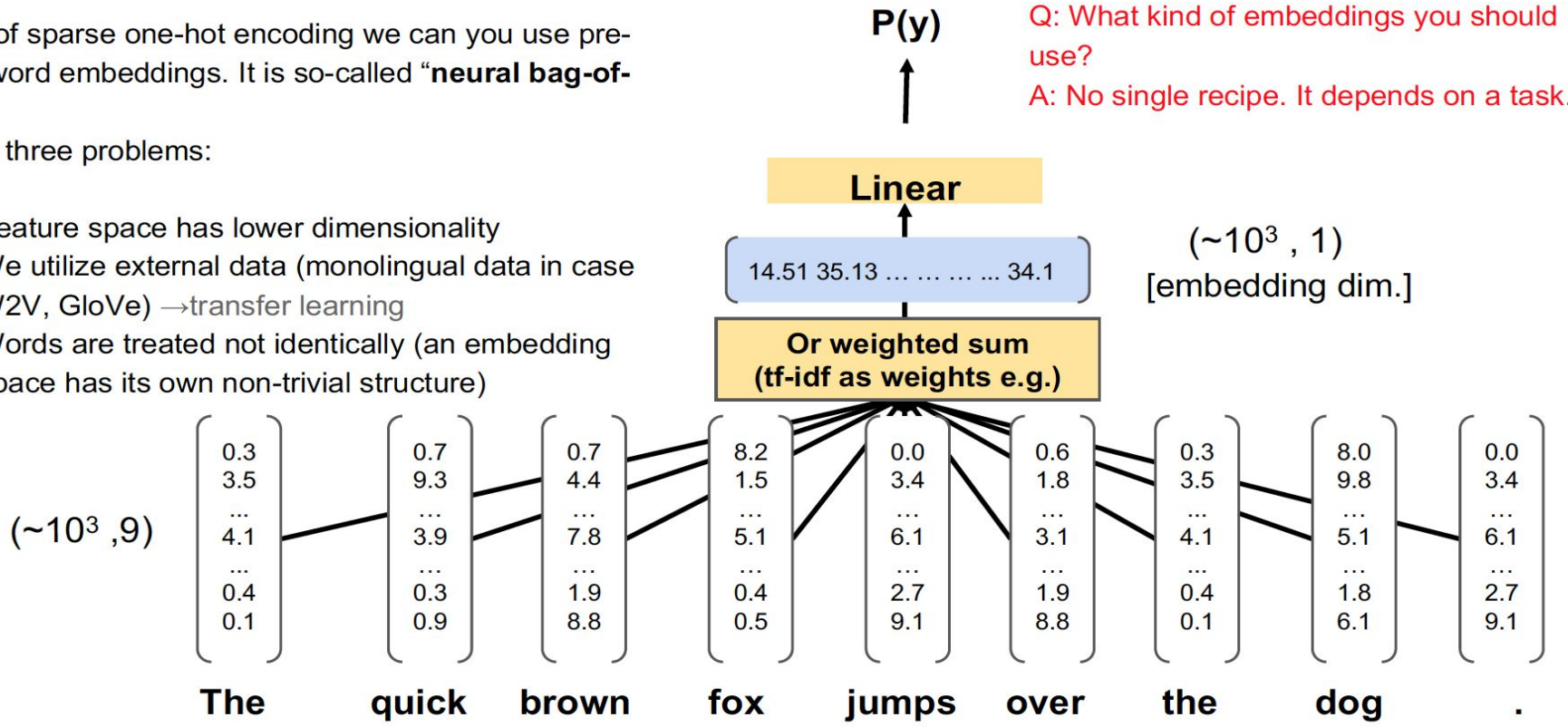


Dense Text Representation: NBOW

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.

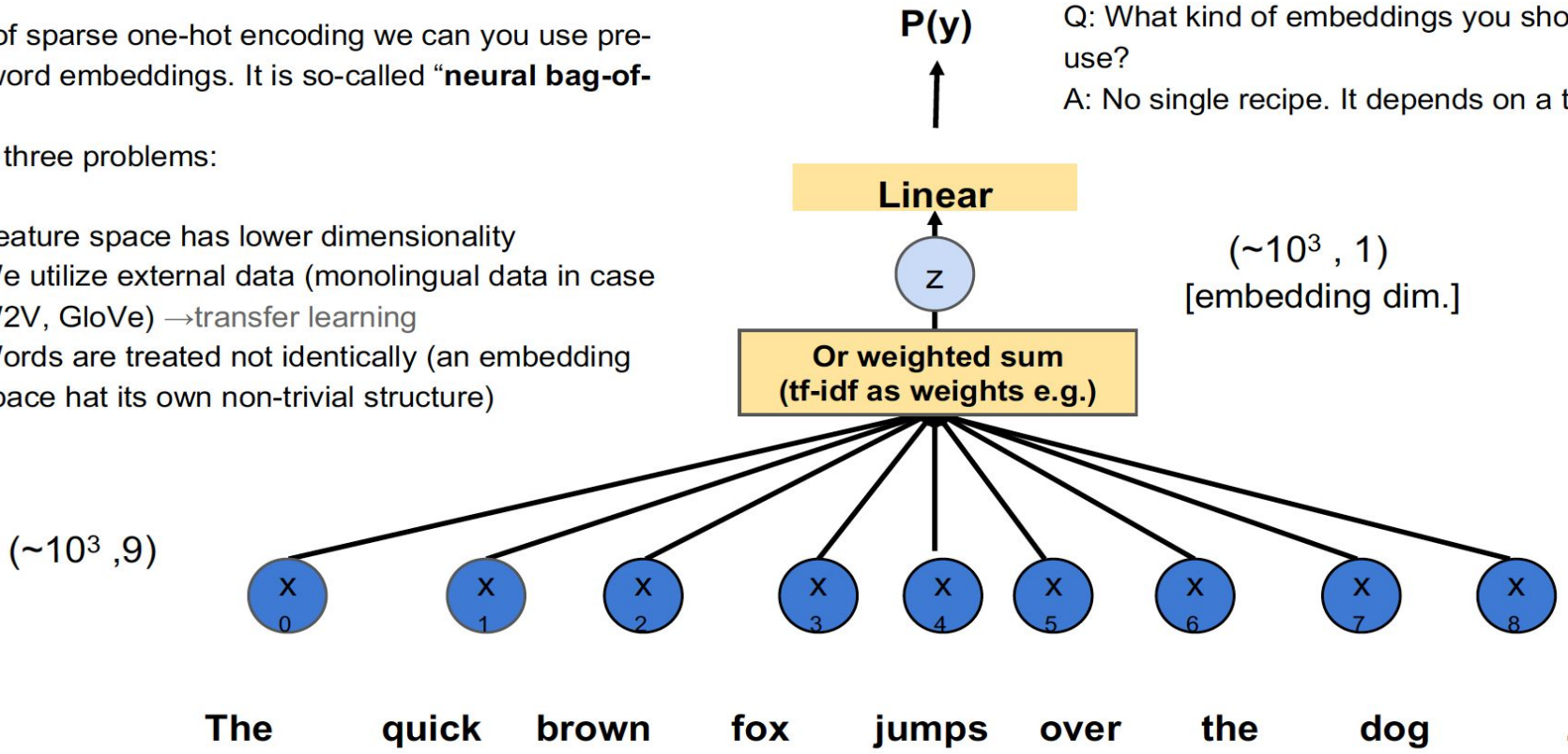


Dense Text Representation: NBOW

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...

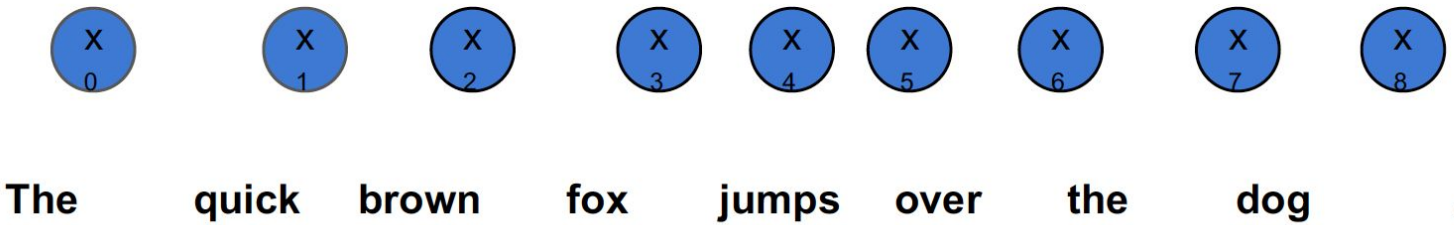
BOW and NBOW: the shared problems

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)

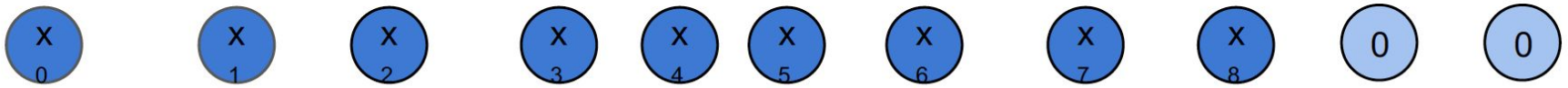
Multilayer perceptron

The simplest learnable aggregation function
is a multilayer perceptron (stacked dense layers).



Multilayer perceptron

The simplest learnable aggregation function
is a multilayer perceptron (stacked dense layers).

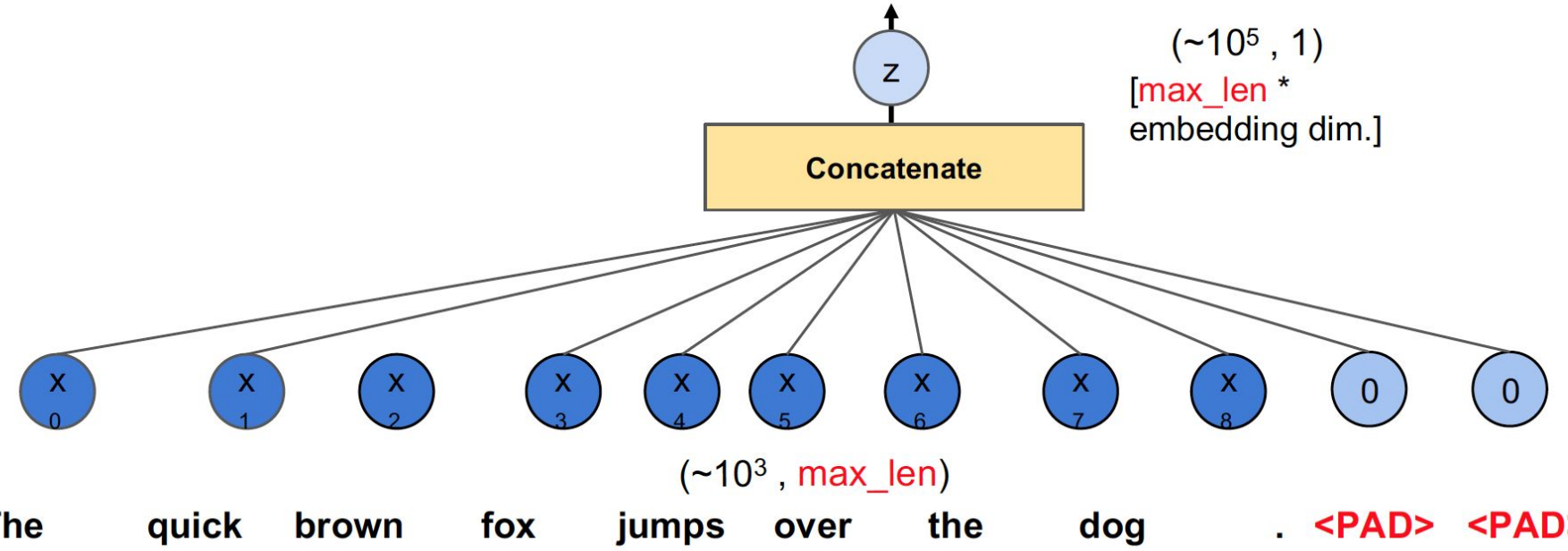


(~10^3 , max_len)

The quick brown fox jumps over the dog . <PAD> <PAD>

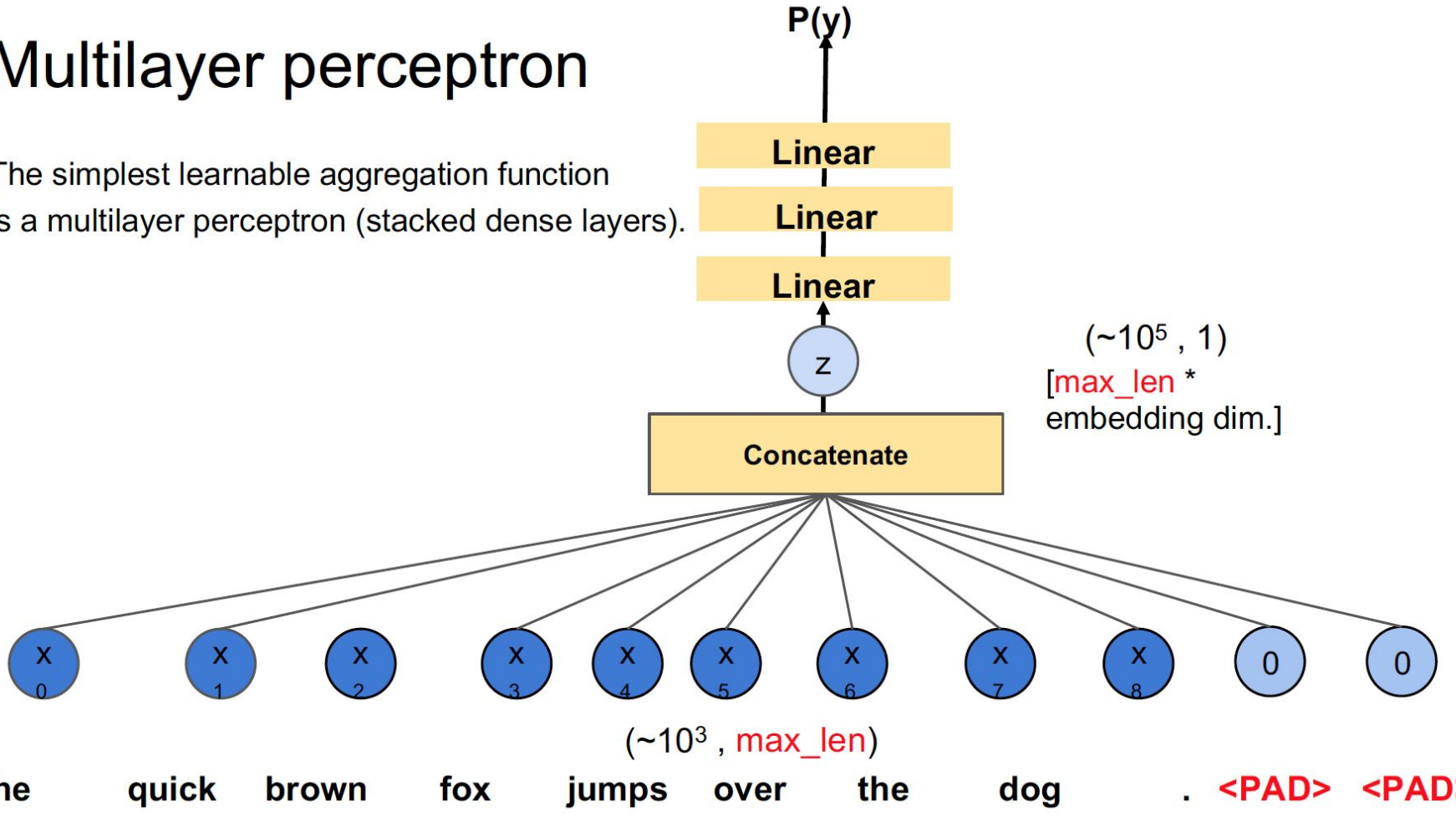
Multilayer perceptron

The simplest learnable aggregation function
is a multilayer perceptron (stacked dense layers).



Multilayer perceptron

The simplest learnable aggregation function
is a multilayer perceptron (stacked dense layers).

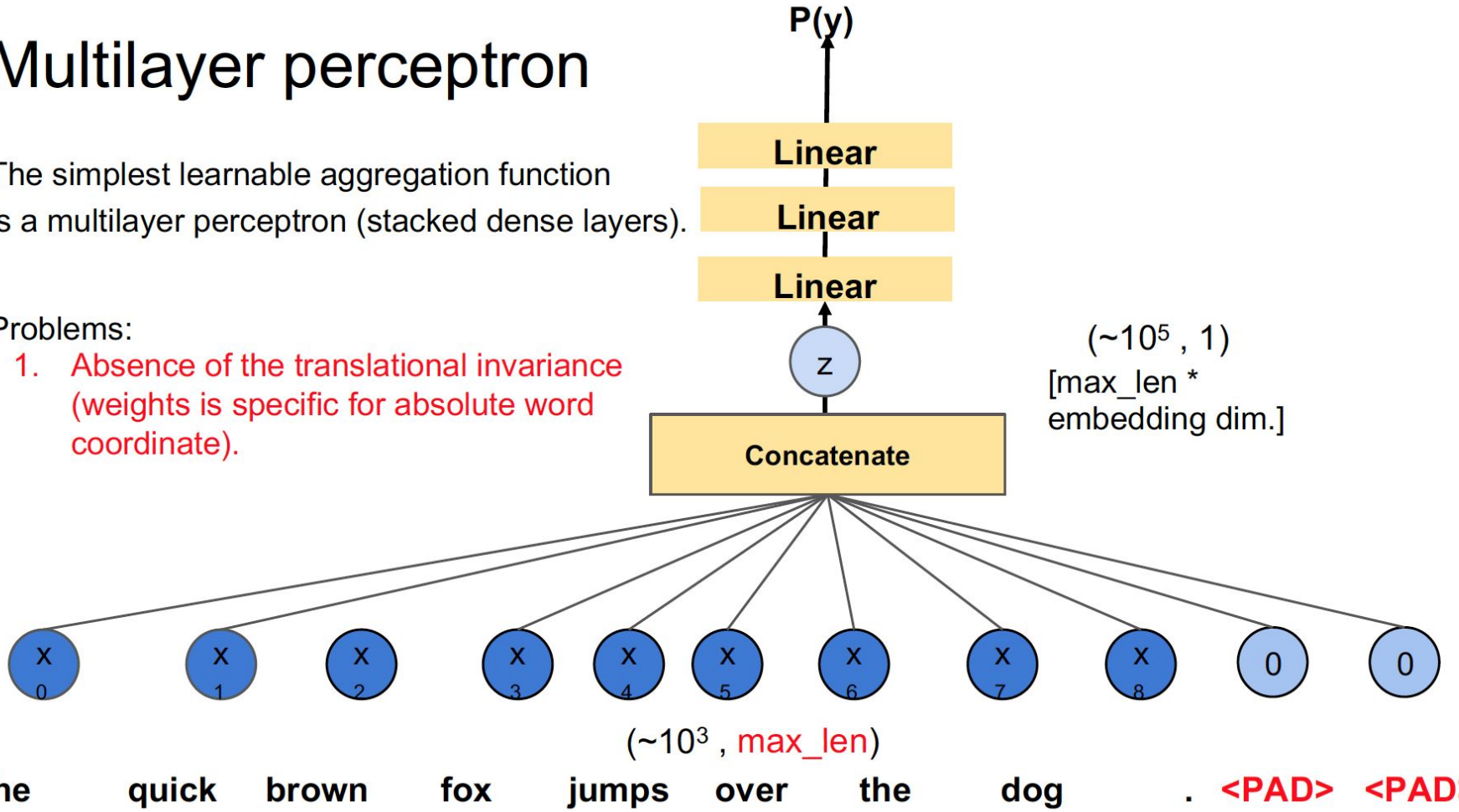


Multilayer perceptron

The simplest learnable aggregation function is a multilayer perceptron (stacked dense layers).

Problems:

1. Absence of the translational invariance (weights is specific for absolute word coordinate).

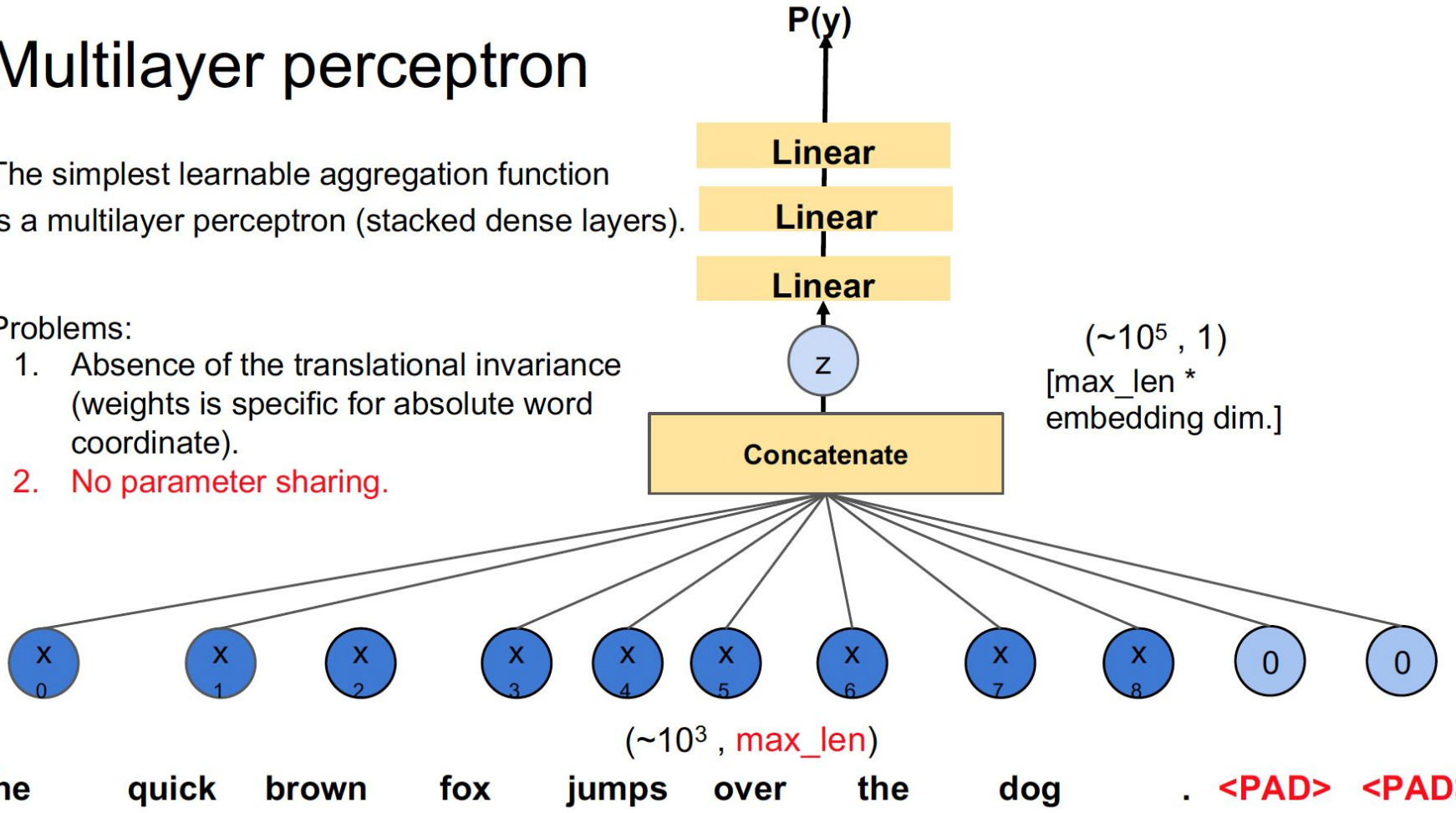


Multilayer perceptron

The simplest learnable aggregation function is a multilayer perceptron (stacked dense layers).

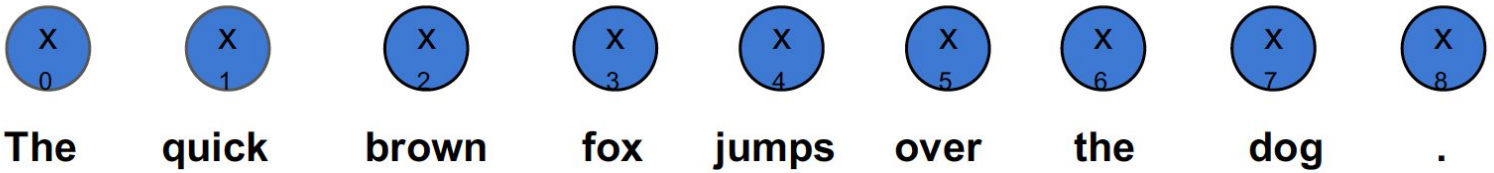
Problems:

1. Absence of the translational invariance (weights is specific for absolute word coordinate).
2. No parameter sharing.



Recurrent NN for text classification

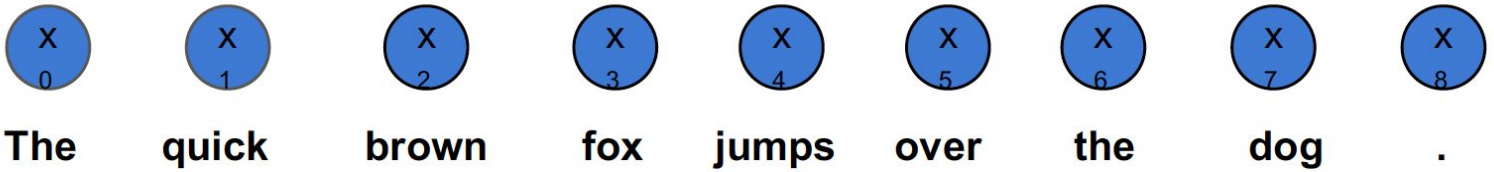
In a RNN Connections between nodes
form a directed graph along a sequence.



Recurrent NN for text classification

In a RNN Connections between nodes
form a directed graph along a sequence.

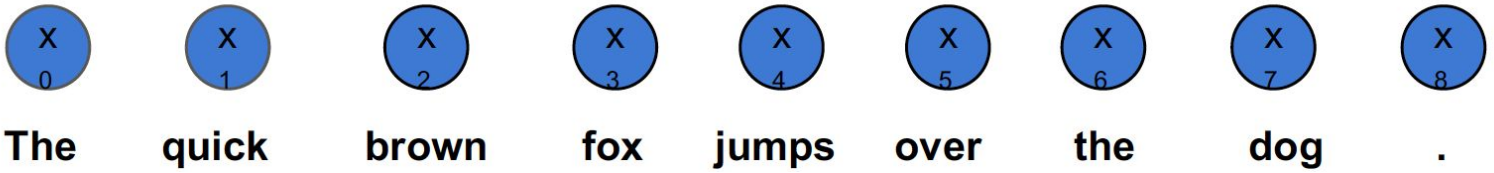
There are different types of recurrent units:
vanilla RNN, LSTM, GRU, MI-LSTM,
peephole LSTM, …
But it's not important this time.



Recurrent NN for text classification

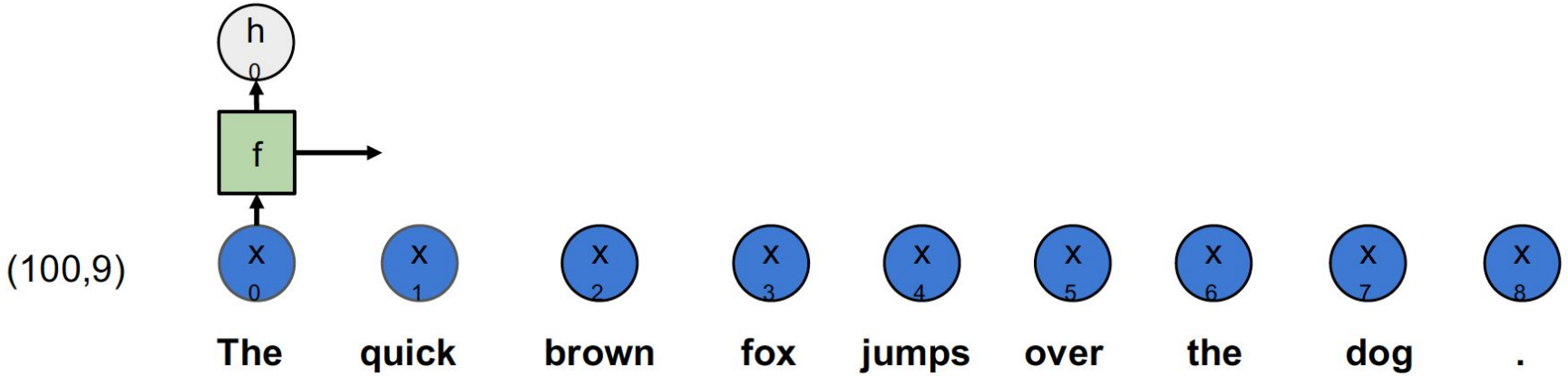
In a RNN Connections between nodes
form a directed graph along a sequence.

$$\mathbf{h}_t = f_w(\mathbf{h}_{t-1}, \mathbf{x}_t)$$



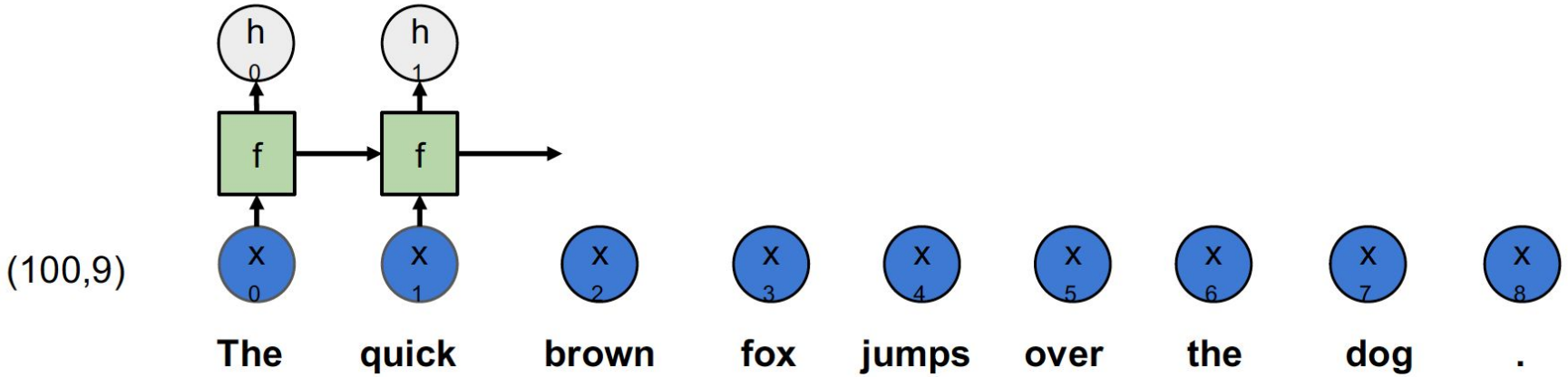
Recurrent NN for text classification

$$\mathbf{h}_t = f_w(\mathbf{h}_{t-1}, \mathbf{x}_t)$$



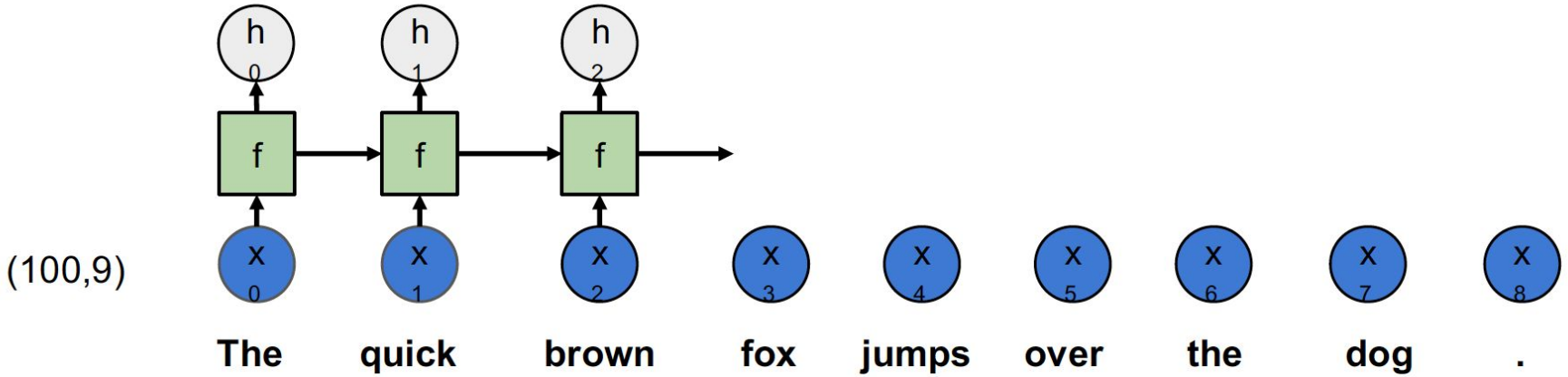
Recurrent NN for text classification

$$\mathbf{h}_t = f_w(\mathbf{h}_{t-1}, \mathbf{x}_t)$$



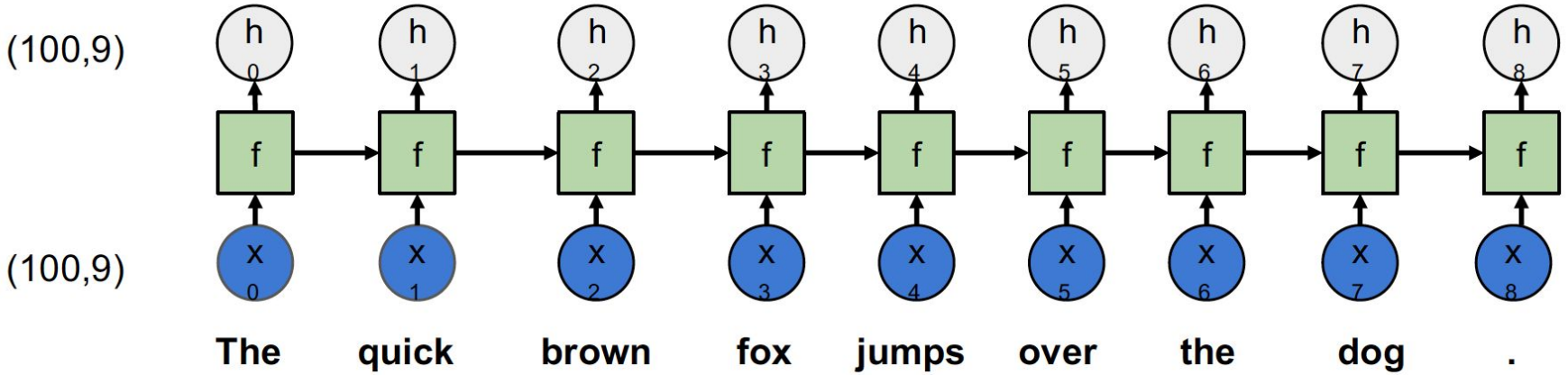
Recurrent NN for text classification

$$\mathbf{h}_t = f_w(\mathbf{h}_{t-1}, \mathbf{x}_t)$$



Recurrent NN for text classification

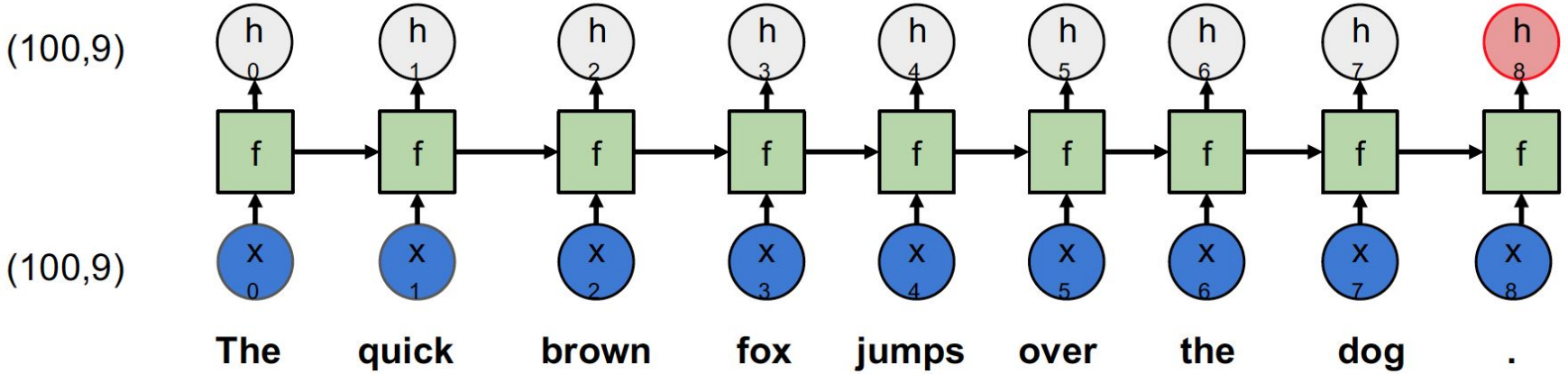
$$\mathbf{h}_t = f_w(\mathbf{h}_{t-1}, \mathbf{x}_t)$$



Recurrent NN for text classification

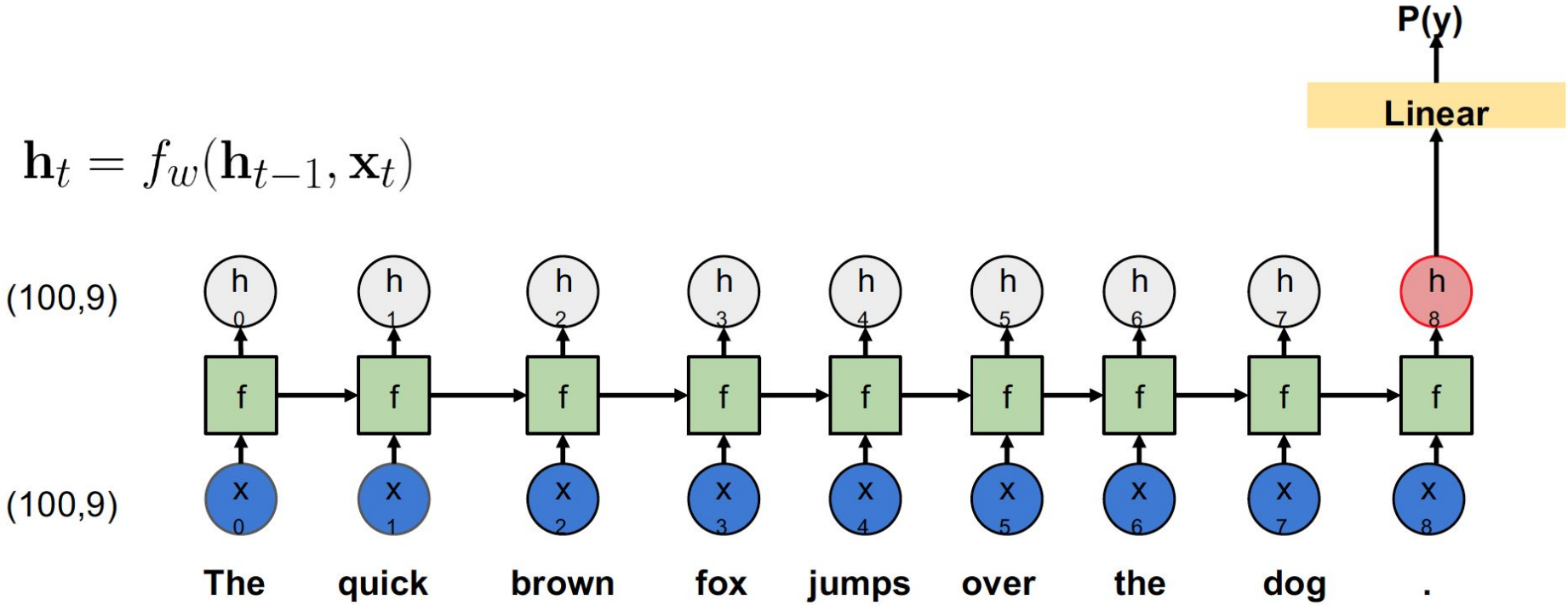
$$\mathbf{h}_8 = f(f(f(...(f(\mathbf{0}, \mathbf{x}_0)), \mathbf{x}_6), \mathbf{x}_7), \mathbf{x}_8)$$

$$\mathbf{h}_t = f_w(\mathbf{h}_{t-1}, \mathbf{x}_t)$$



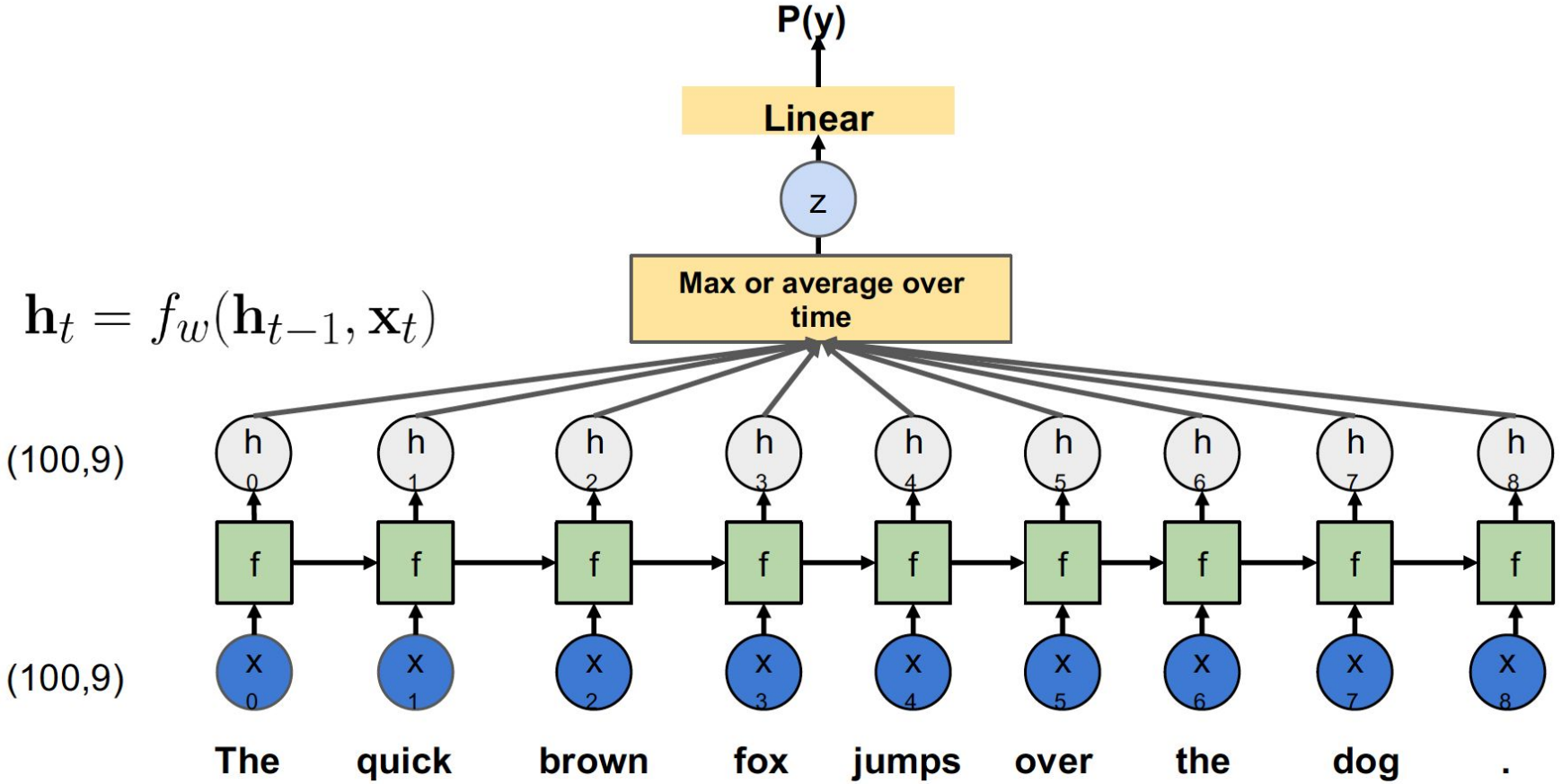
Recurrent NN for text classification

$$\mathbf{h}_t = f_w(\mathbf{h}_{t-1}, \mathbf{x}_t)$$



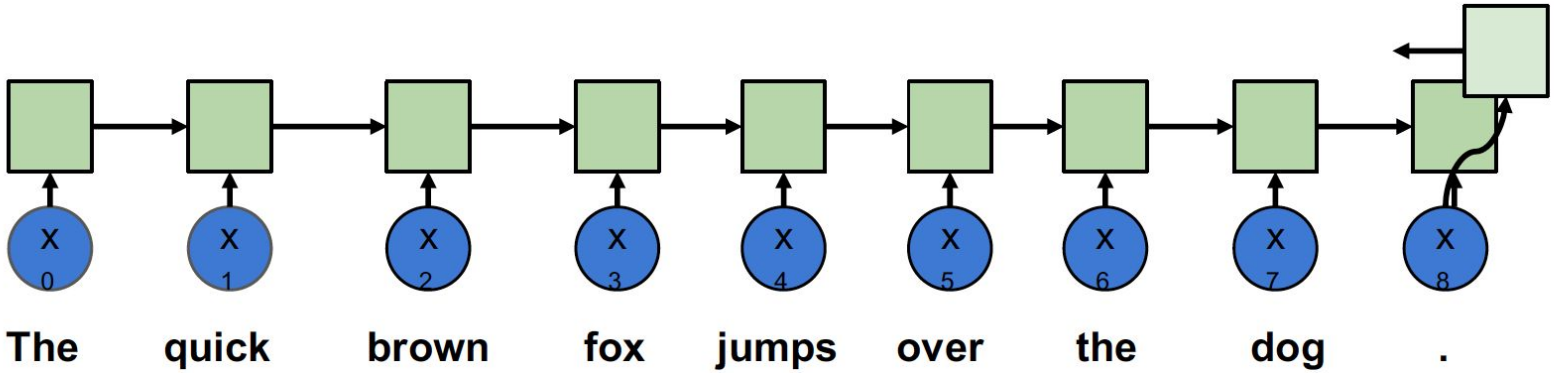
Recurrent NN for text classification

$$\mathbf{h}_t = f_w(\mathbf{h}_{t-1}, \mathbf{x}_t)$$



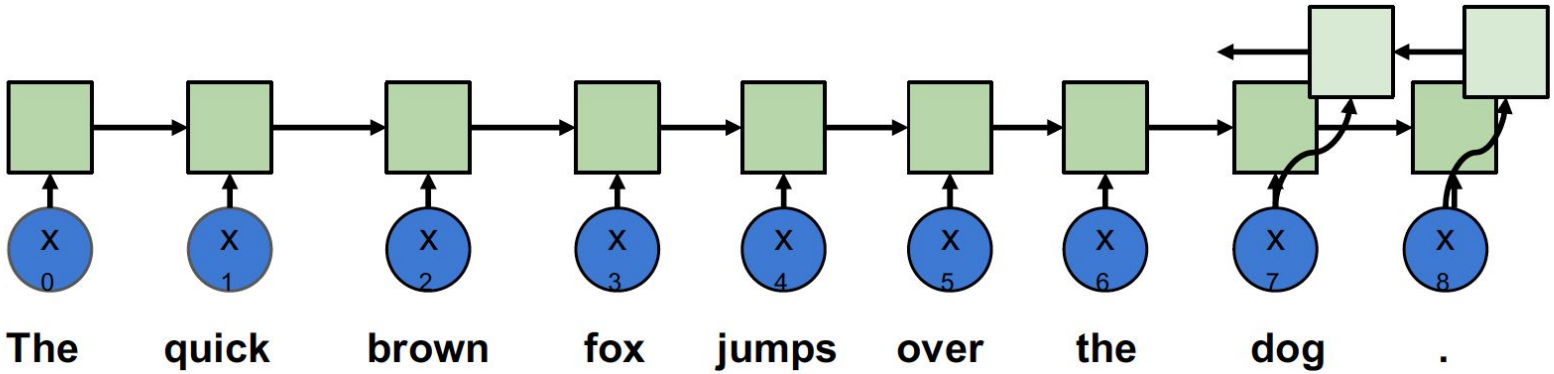
Recurrent NN for text classification

(100,9)



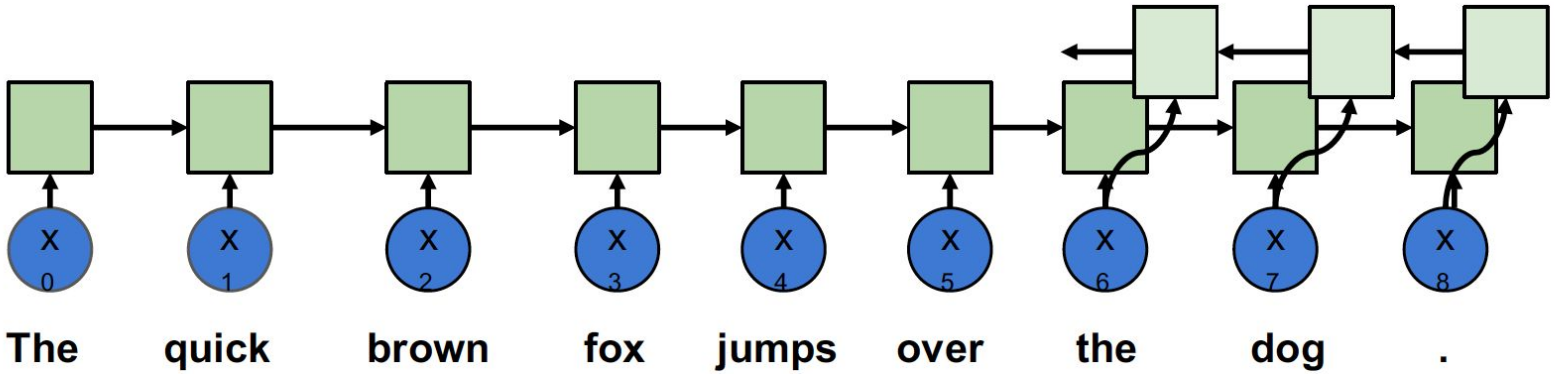
Recurrent NN for text classification

(100,9)



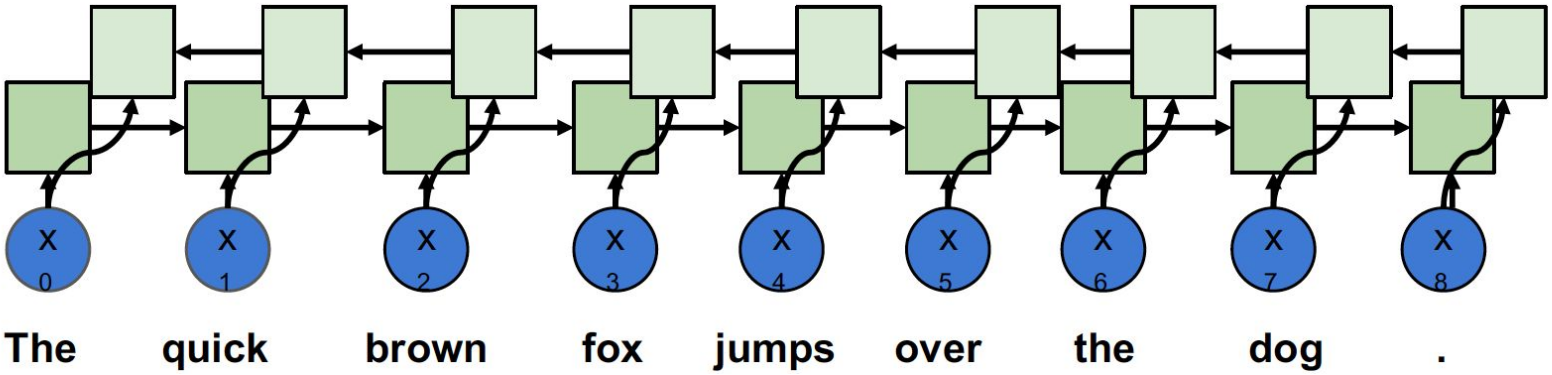
Recurrent NN for text classification

(100,9)

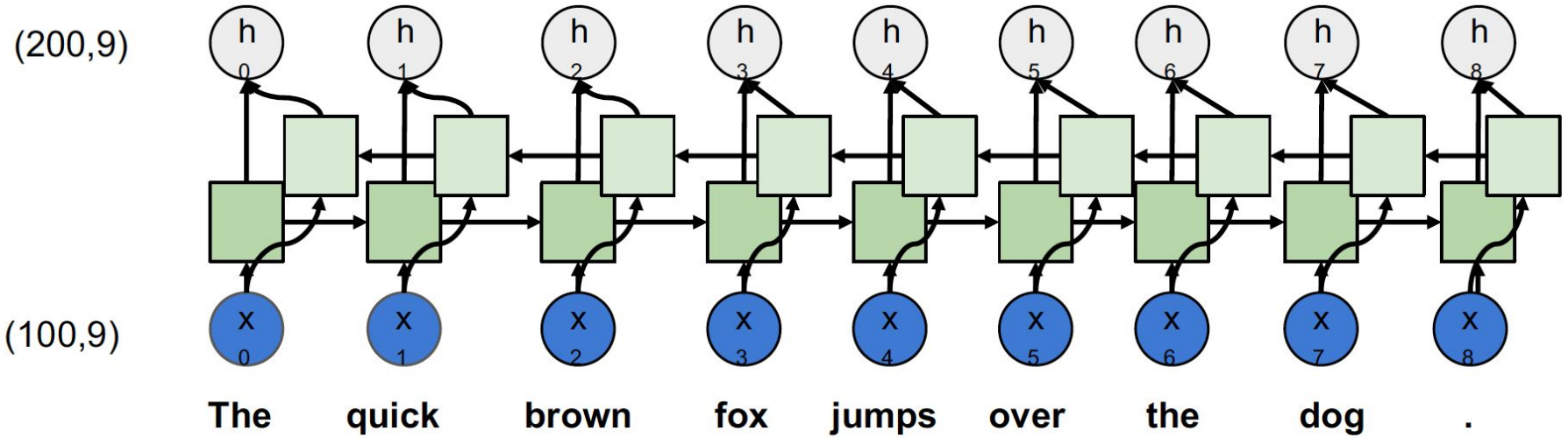


Recurrent NN for text classification

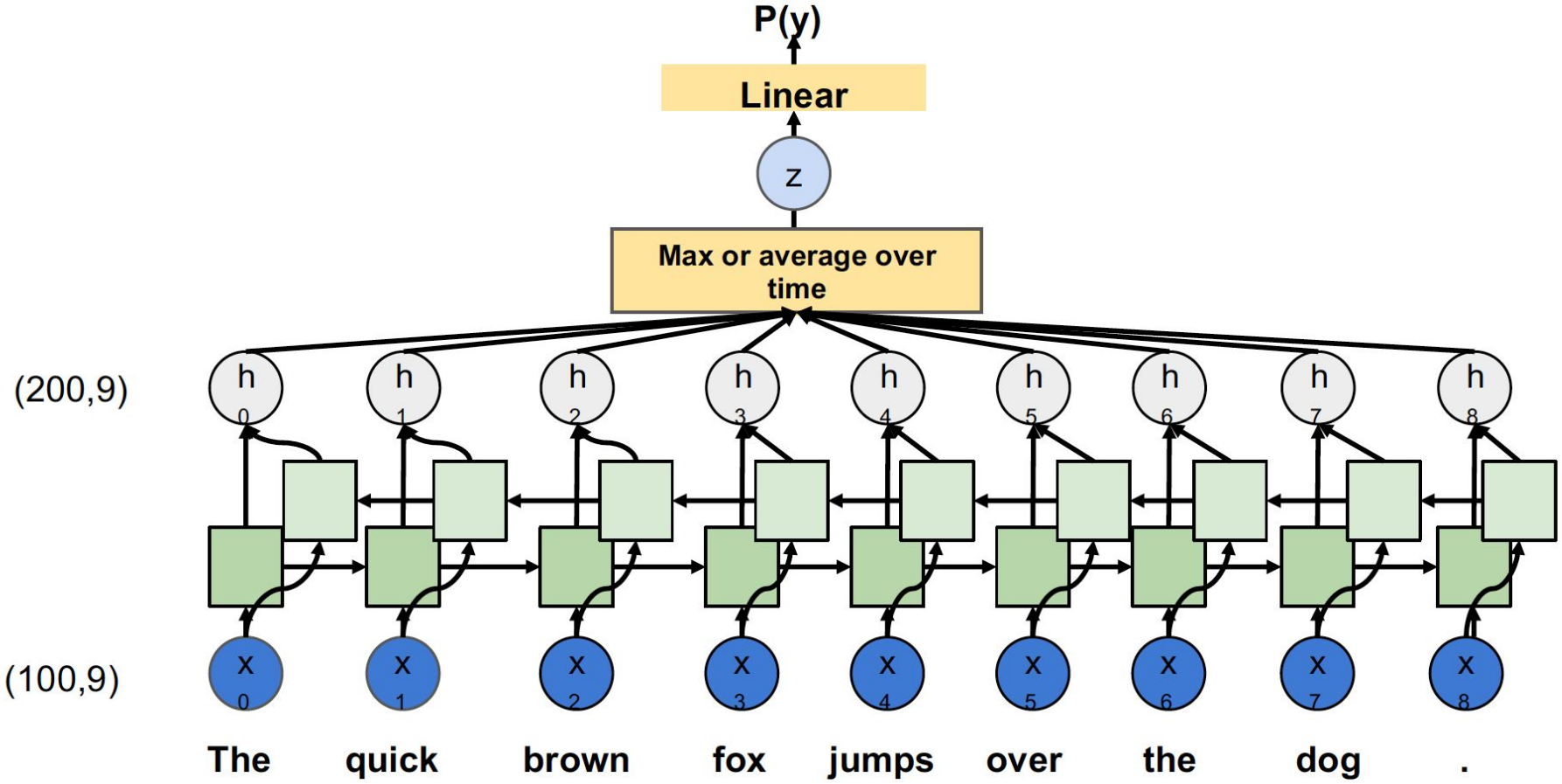
(100,9)



Recurrent NN for text classification



Recurrent NN for text classification



References

- CS224n: Natural Language Processing with Deep Learning
- NLP course by Yandex data school
- https://medium.com/data-from-the-trenches/text-classification-the-first-step-toward-nlp-mastery-f5f95d525d73

Thank You!

The Hitchhiker's Guide to the Galaxy *by* Douglas Adams's

The Hitchhiker's Guide to the Intelligence ™!

Don’t Panic Guide to AI