Text Classification

September 20, 2018

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?

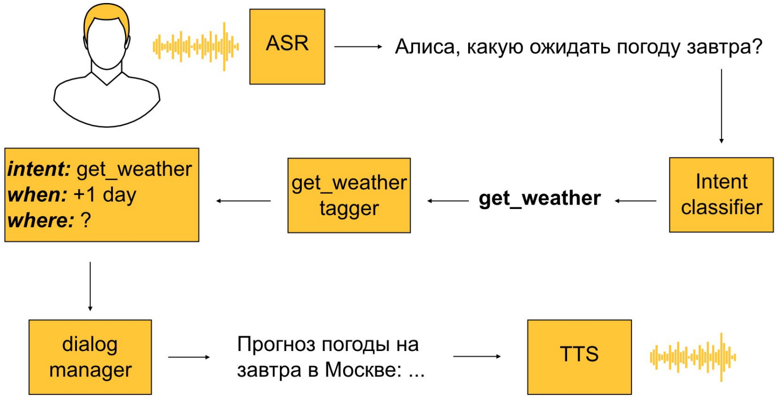
- As a self-sufficient task:
  - Spam filtering
  - Sentiment analysis
  - Fake news/clickbait detection
  - Troll/bot protection
- As a part of more complicated NLP tasks

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

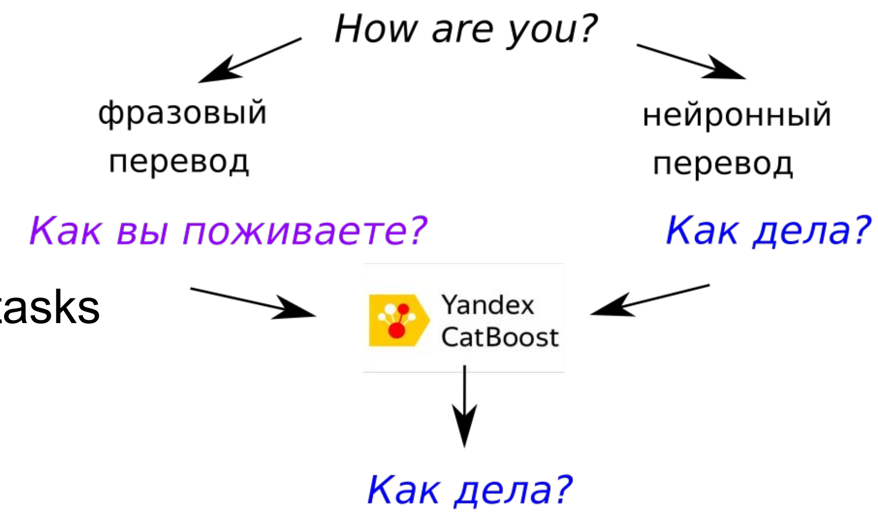
Why do we need to classify texts?

- 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



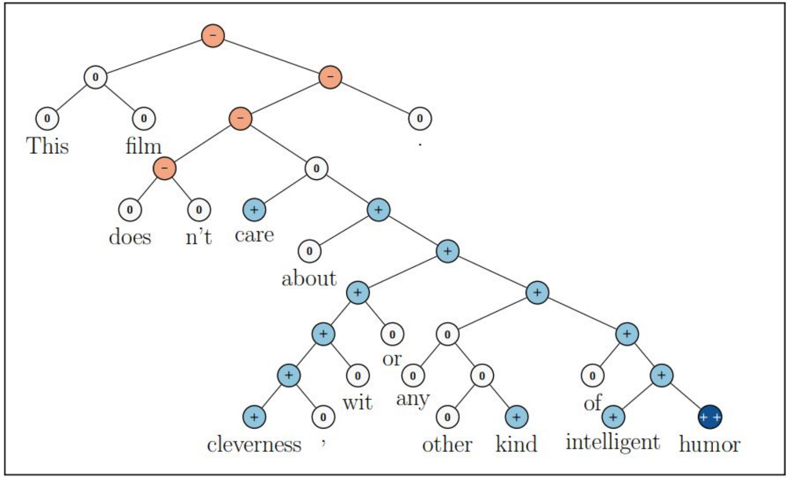
Why do we need to classify texts?

- 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

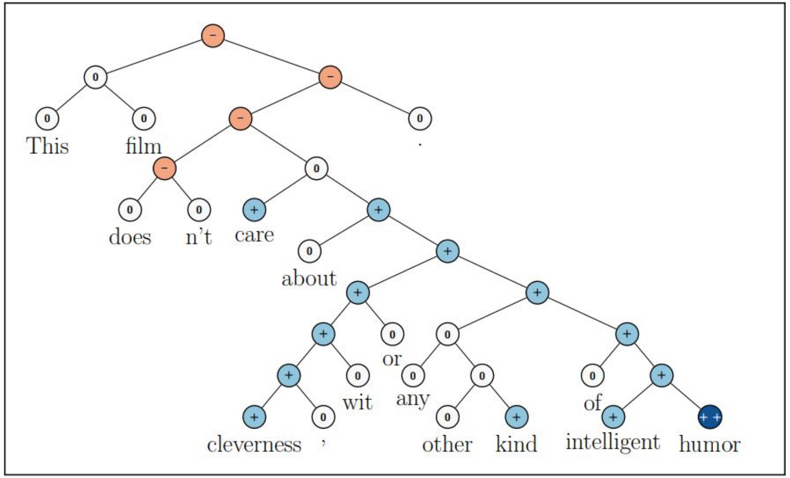


Subjective unigram	**Objective unigram**
amazing, beautiful, cheap, decent, effective, fantastic, good, happy, impress, jittery, light, madly, nice, outstanding, perfect, quick, responsive, sharp, terrible, ultimate, wonderful.	access, because, chance, default, entire, few, go, half, inside, job, keep,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

Popular Benchmarks

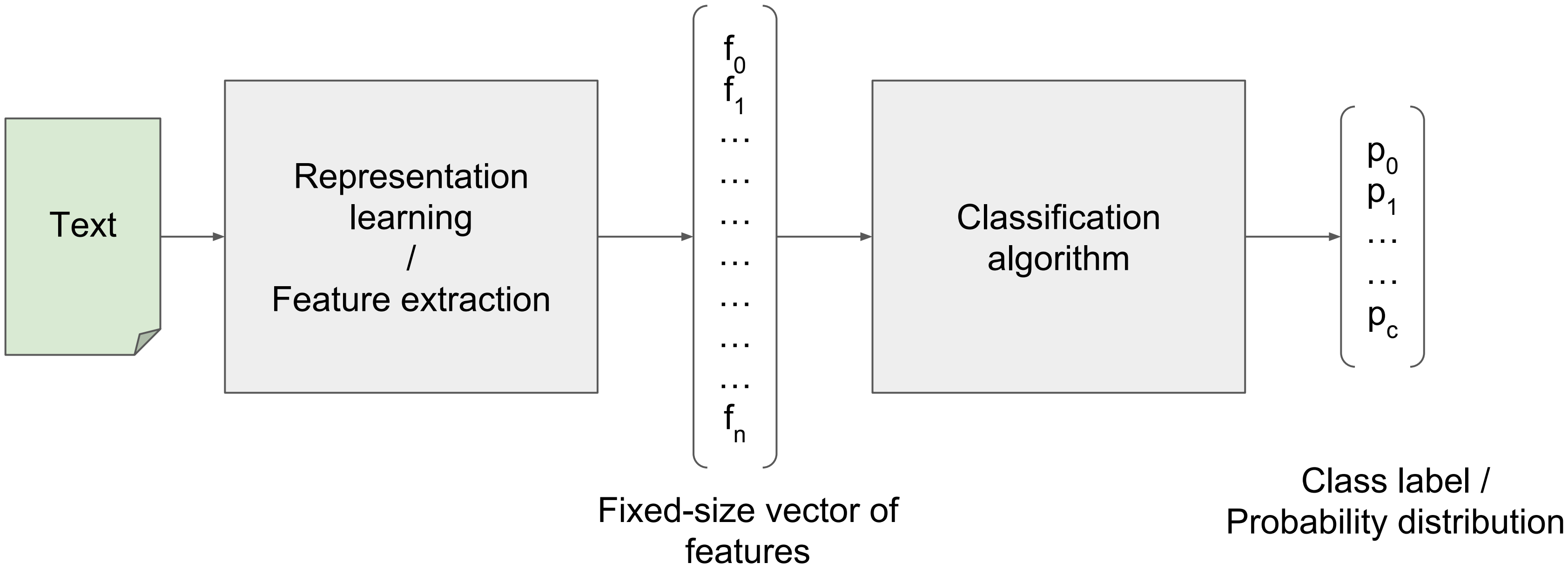
Dataset	Size
Question classification (TREC)	6K
MPQA Opinion corpus (SUBJ)	8K
Movie Reviews (MR)	10K
Reuters-21578	21.5K
IMDB Reviews	25K
Stanford Sentiment Treebank (SST)	9.5K
Sogou News	0.5M
AG News	1M
Yelp Dataset	5.2M
Amazon Reviews	35M
Flickr 100m	100M

~2012

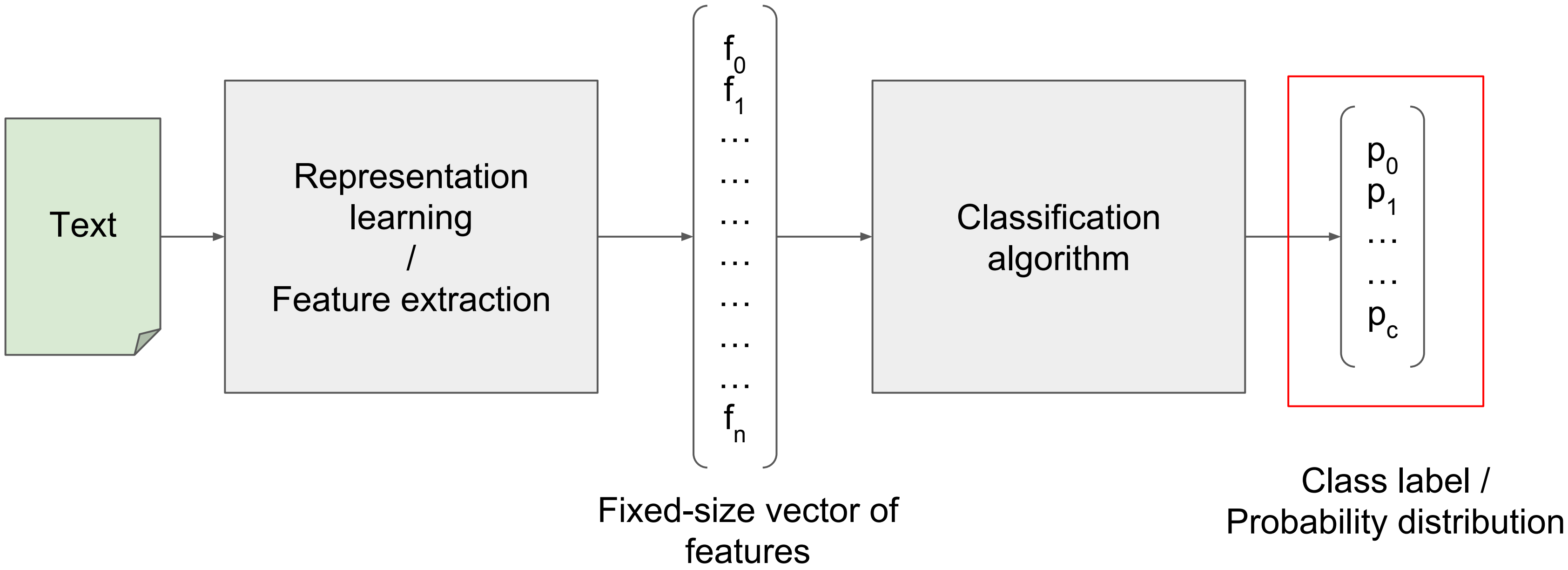


Subjective unigram	Objective unigram
amazing, beautiful, cheap, decent, effective, fantastic, good, happy, impress, jittery, light, madly, nice, outstanding, perfect, quick, responsive, sharp, terrible, ultimate, wonderful.	access, because, chance, default, entire, few, go, half, inside, job, keep, know,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

Text classification in general



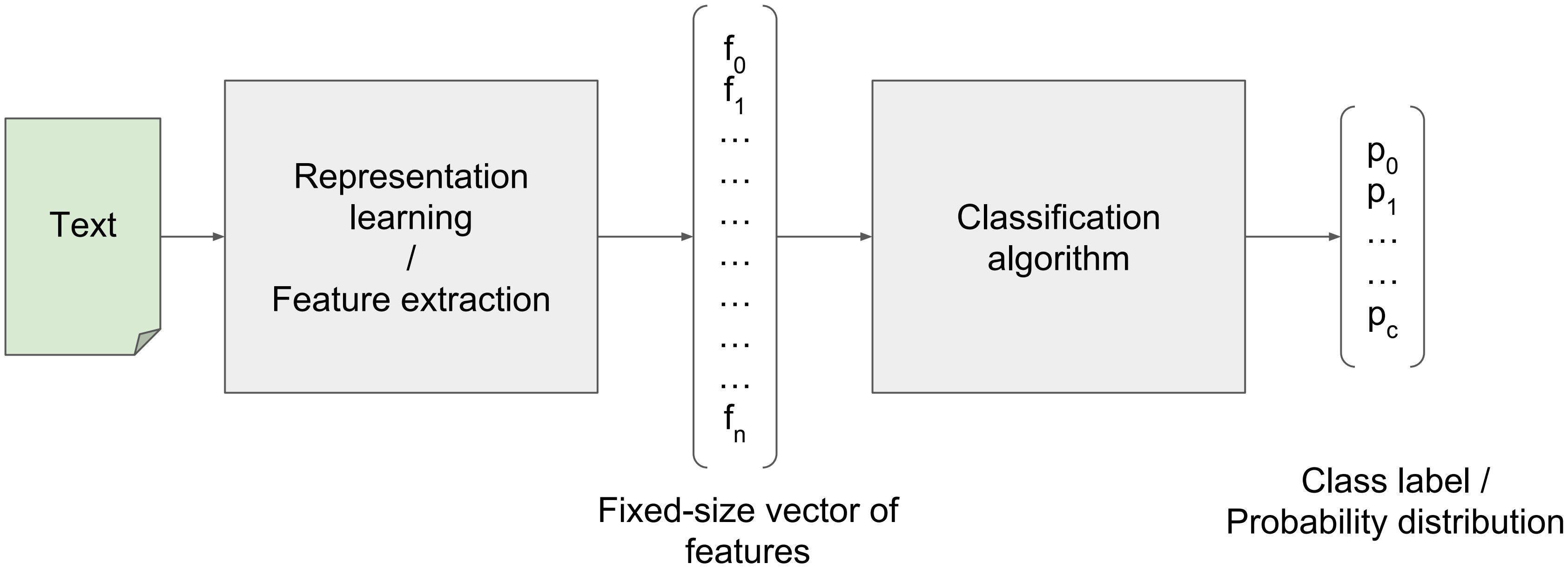
Text classification in general



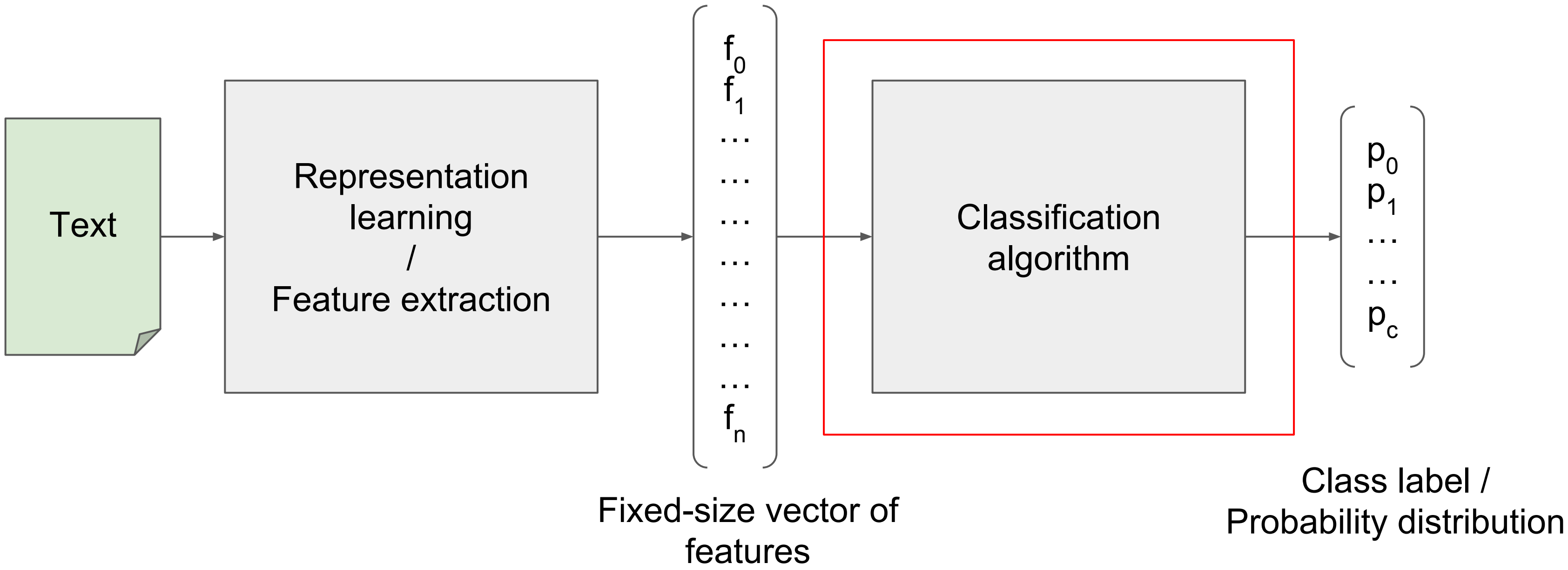
Text label kinds

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



Text classification in general



Some words about classification algorithms

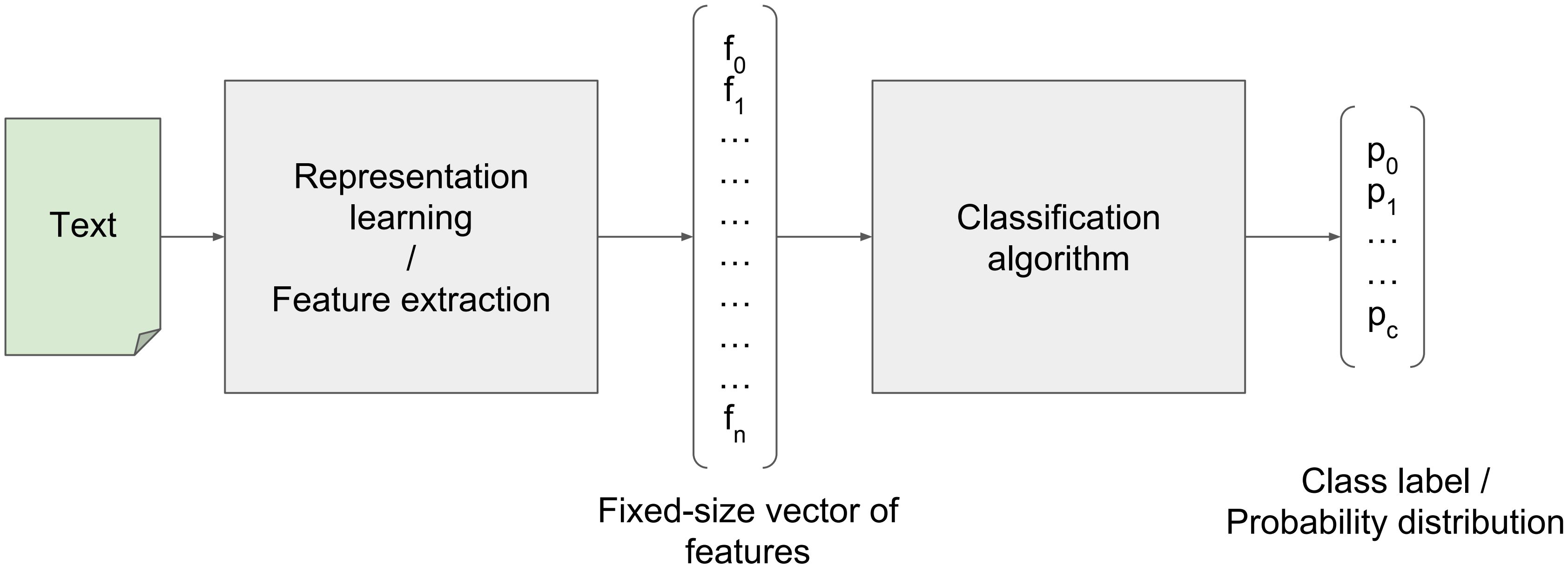
NLP has its own cozy atmosphere when it comes to naming models

TL;DR

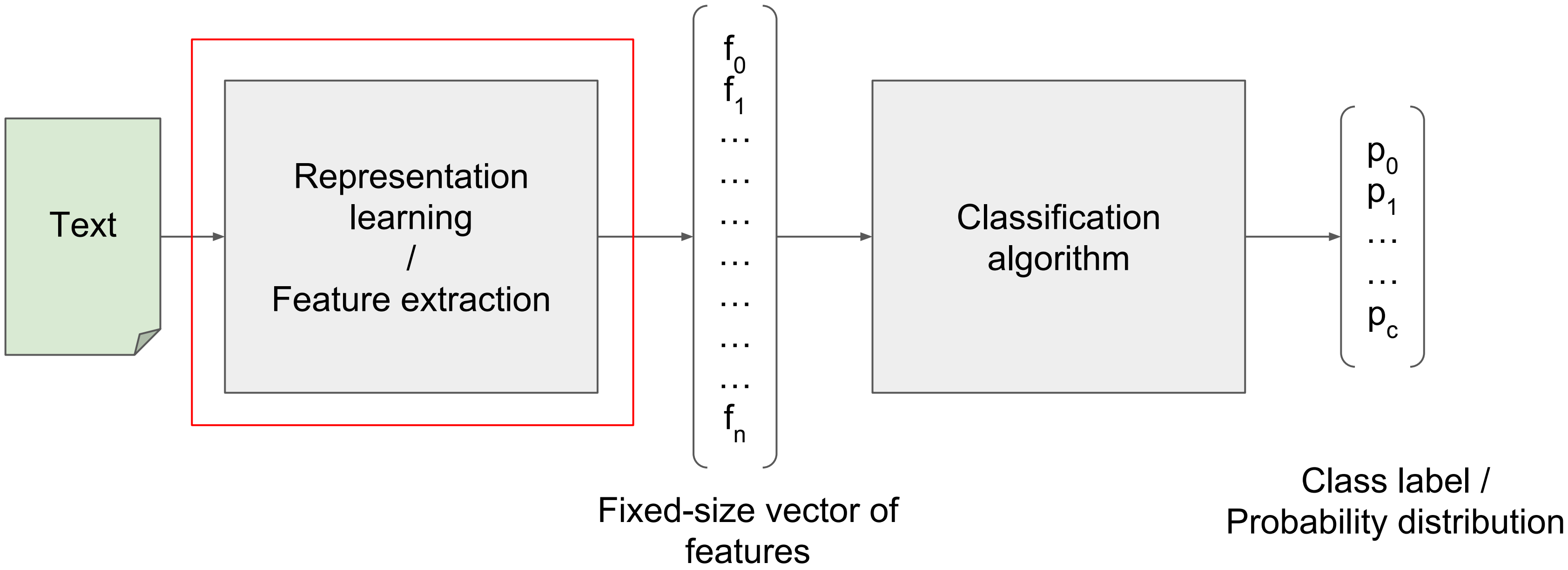
Maximum Entropy Method = sklearn.linear_model.LogisticRegression

Multinomial Naive Bayes = sklearn.naive_bayes.MultinomialNB

Text classification in general



Text classification in general



Text representation: feature engineering

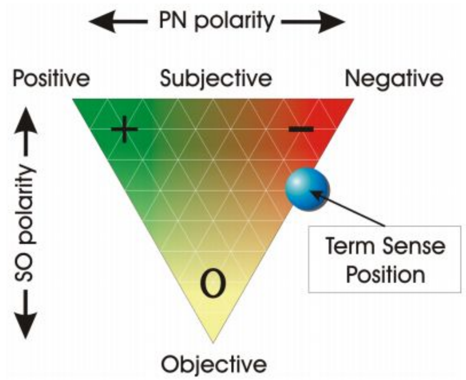
As for many ML tasks, it is possible to generate useful features by hands.

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**	**~~lazy~~** **the**	**dog**	**.**

Sparse text representation: BOW

The quick brown fox jumps over the dog .

Sparse text representation: BOW

$\begin{pmatrix} 1 \\ 0 \\ \cdots \\ 0 \\ \cdots \\ 0 \end{pmatrix}$						$\begin{pmatrix} 1 \\ 0 \\ \cdots \\ \ldots \\ \cdots \\ 0 \end{pmatrix}$		
The	**quick**	**brown**	**fox**	**jumps**	**over**	**the**	**dog**	**.**

Sparse text representation: BOW

$$\begin{pmatrix}1\\0\\\cdots\\0\\\cdots\\0\end{pmatrix}\quad\begin{pmatrix}0\\1\\\cdots\\0\\\cdots\\0\end{pmatrix}\quad\begin{pmatrix}1\\0\\\cdots\\\ldots\\\cdots\\0\end{pmatrix}$$

The **quick** **brown** **fox** **jumps** **over** **the** **dog** **.**

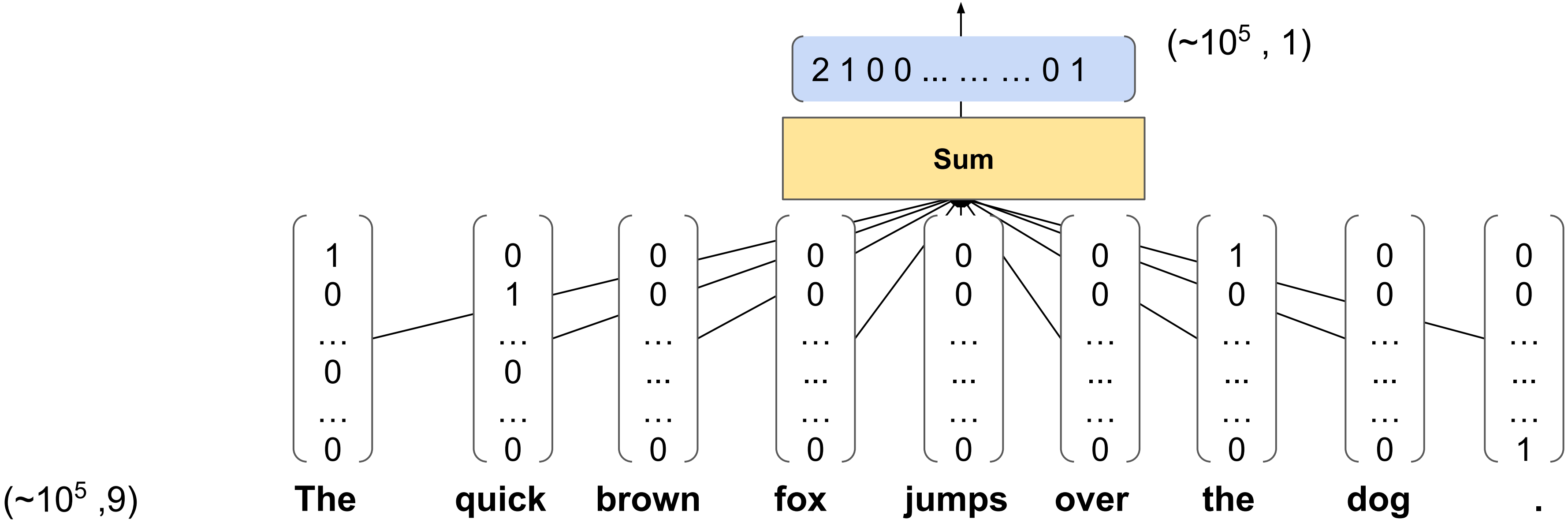
Sparse text representation: BOW

$$
\begin{array}{ccccccccc}
\begin{pmatrix} 1 \\ 0 \\ \cdots \\ 0 \\ \cdots \\ 0 \end{pmatrix} &
\begin{pmatrix} 0 \\ 1 \\ \cdots \\ 0 \\ \cdots \\ 0 \end{pmatrix} &
\begin{pmatrix} 0 \\ 0 \\ \cdots \\ \ldots \\ \cdots \\ 0 \end{pmatrix} &
 & & &
\begin{pmatrix} 1 \\ 0 \\ \cdots \\ \ldots \\ \cdots \\ 0 \end{pmatrix} & & \\
\textbf{The} & \textbf{quick} & \textbf{brown} & \textbf{fox} & \textbf{jumps} & \textbf{over} & \textbf{the} & \textbf{dog} & \textbf{.}
\end{array}
$$

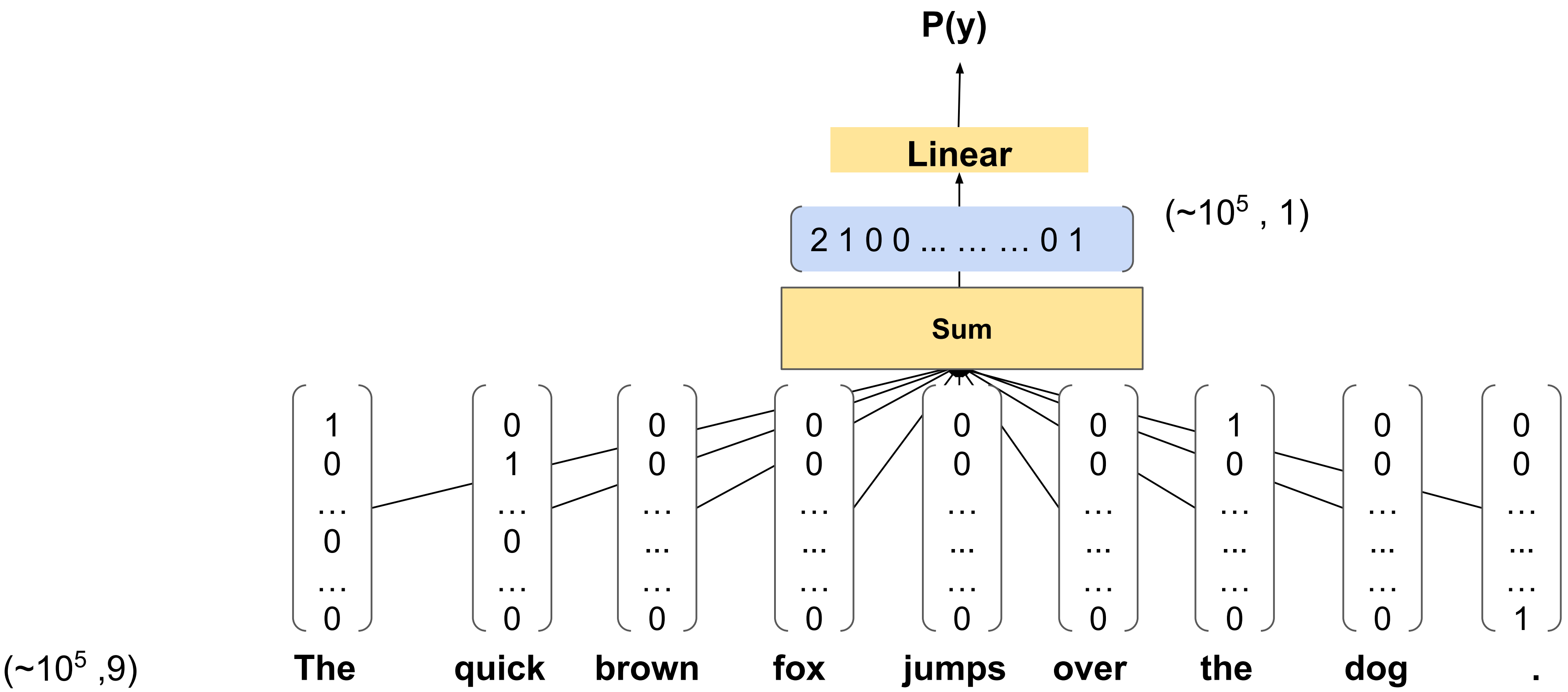
Sparse text representation: BOW

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

Sparse text representation: BOW

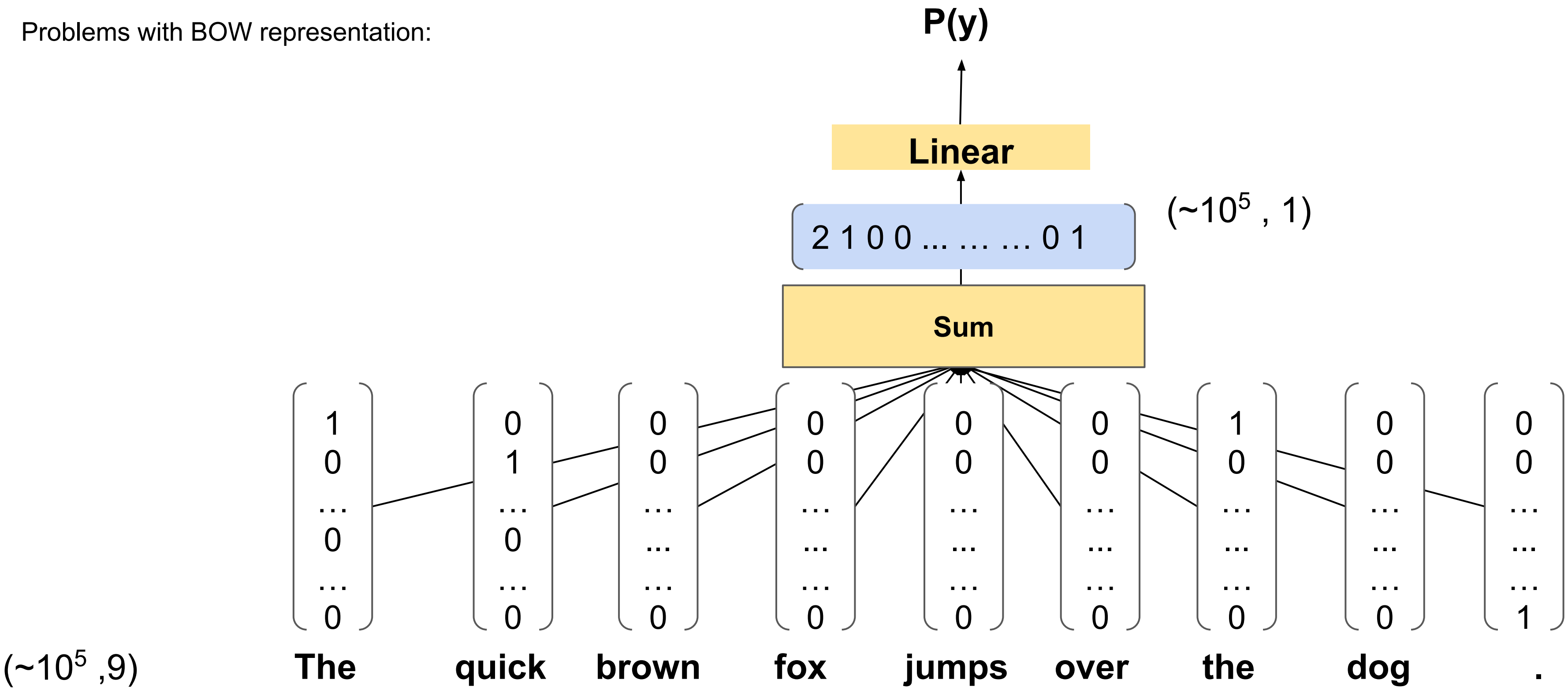


Sparse text representation: BOW



Sparse text representation: BOW

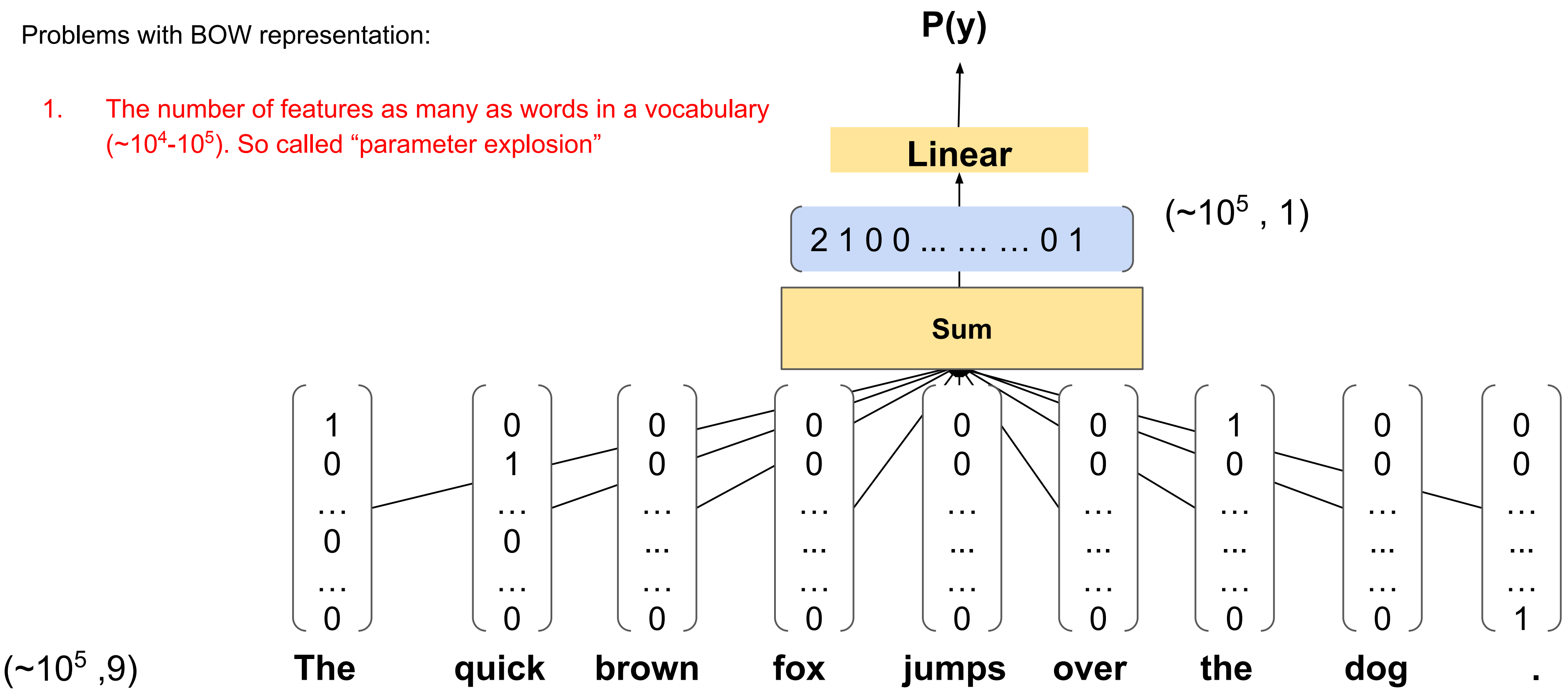
Problems with BOW representation:



Sparse text representation: BOW

Problems with BOW representation:

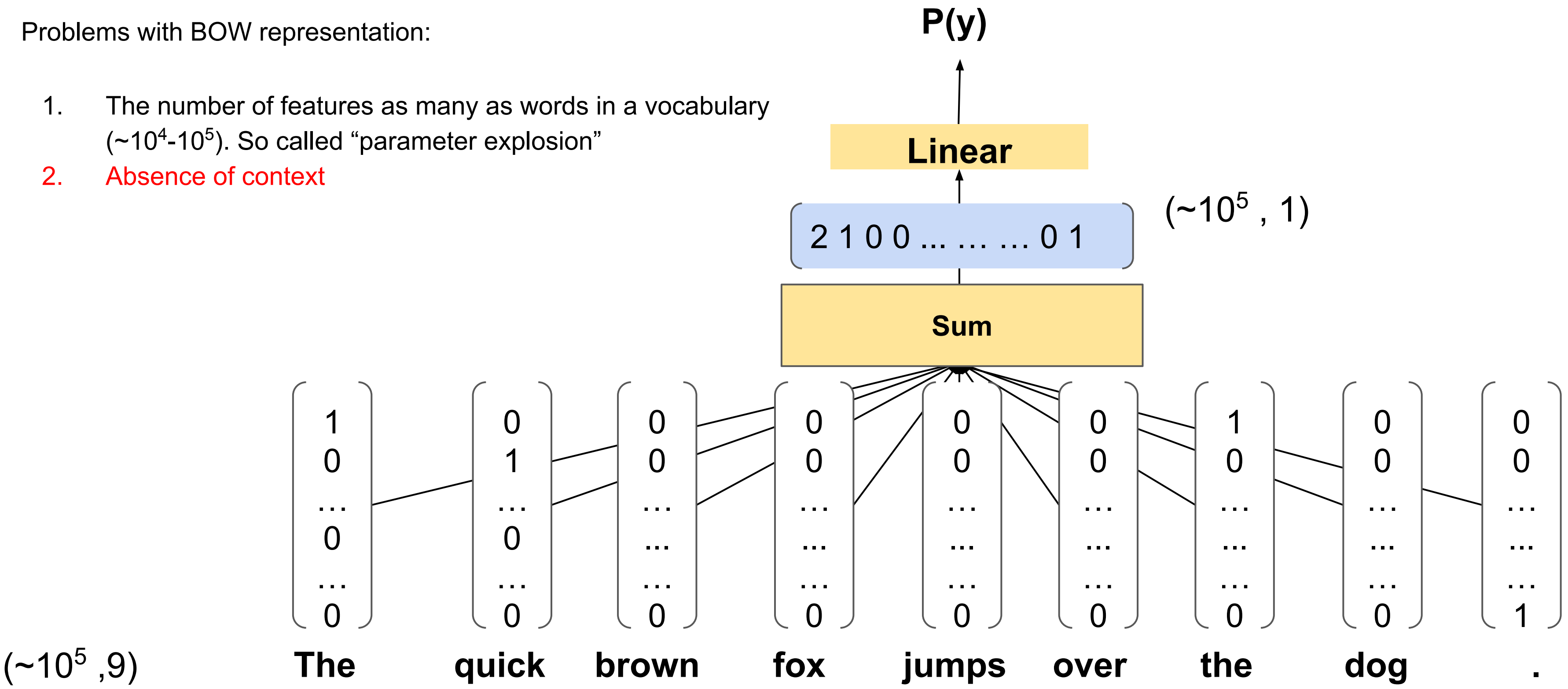
1. The number of features as many as words in a vocabulary ($\sim 10^4$-10^5). So called “parameter explosion”



Sparse text representation: BOW

Problems with BOW representation:

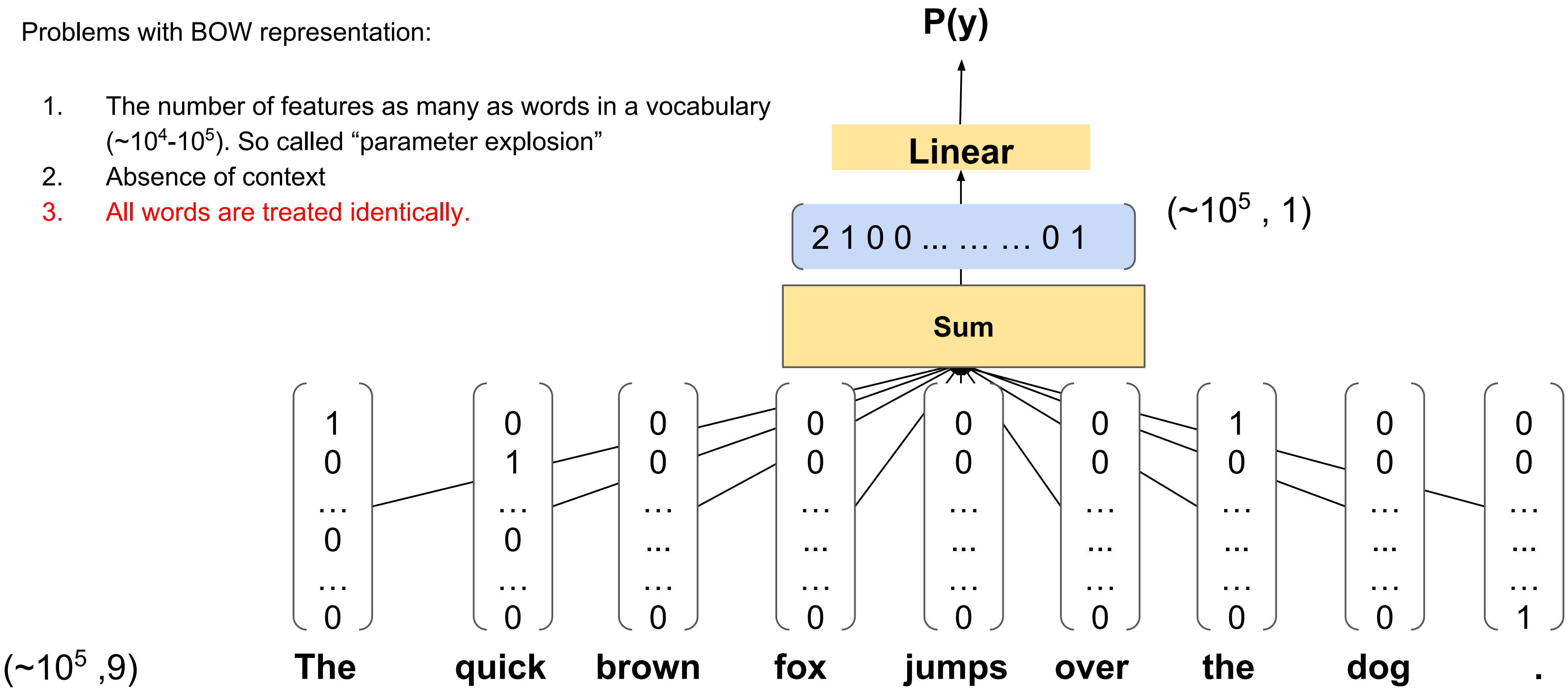
1. The number of features as many as words in a vocabulary ($\sim 10^4$-10^5). So called "parameter explosion"
2. Absence of context



Sparse text representation: BOW

Problems with BOW representation:

1. The number of features as many as words in a vocabulary ($\sim10^4$-10^5). So called “parameter explosion”
2. Absence of context
3. All words are treated identically.

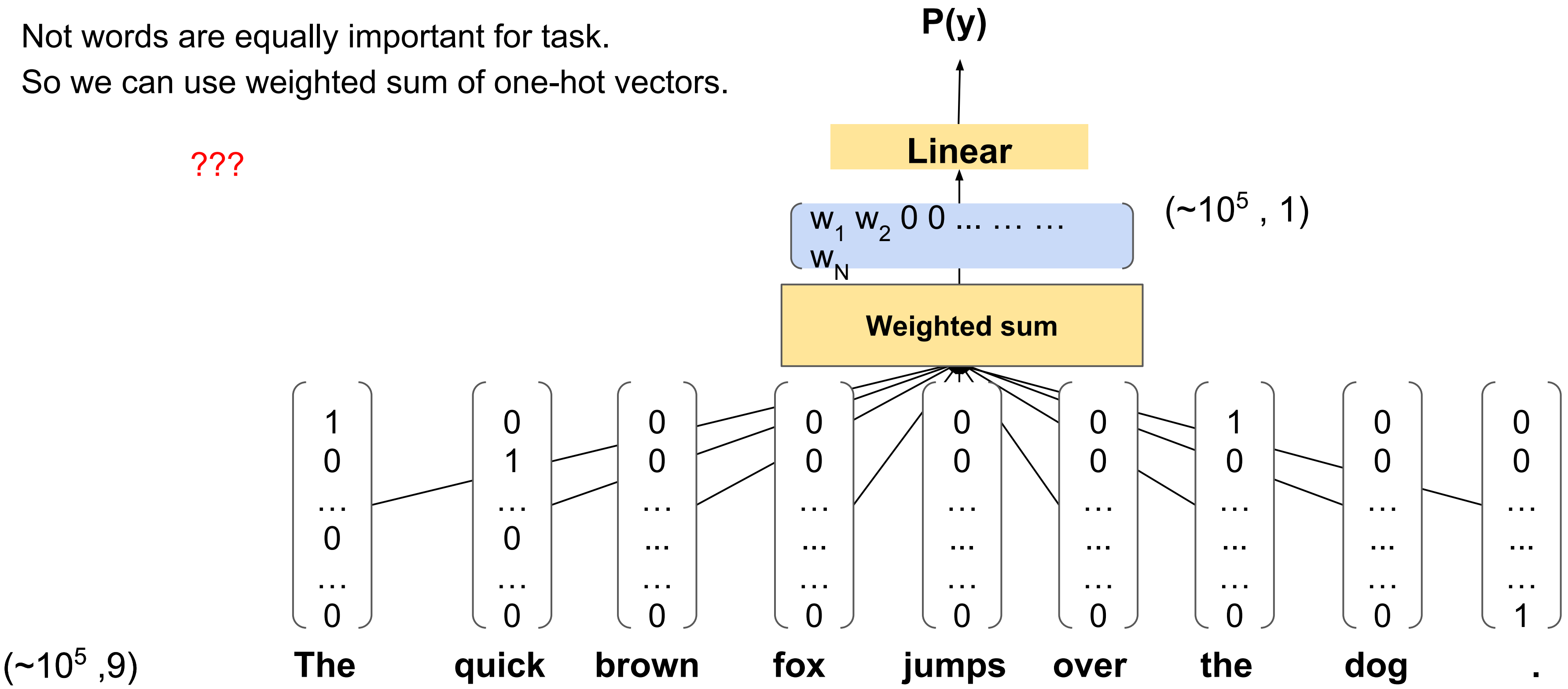


Weighting techniques for BOW

Not words are equally important for task.

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

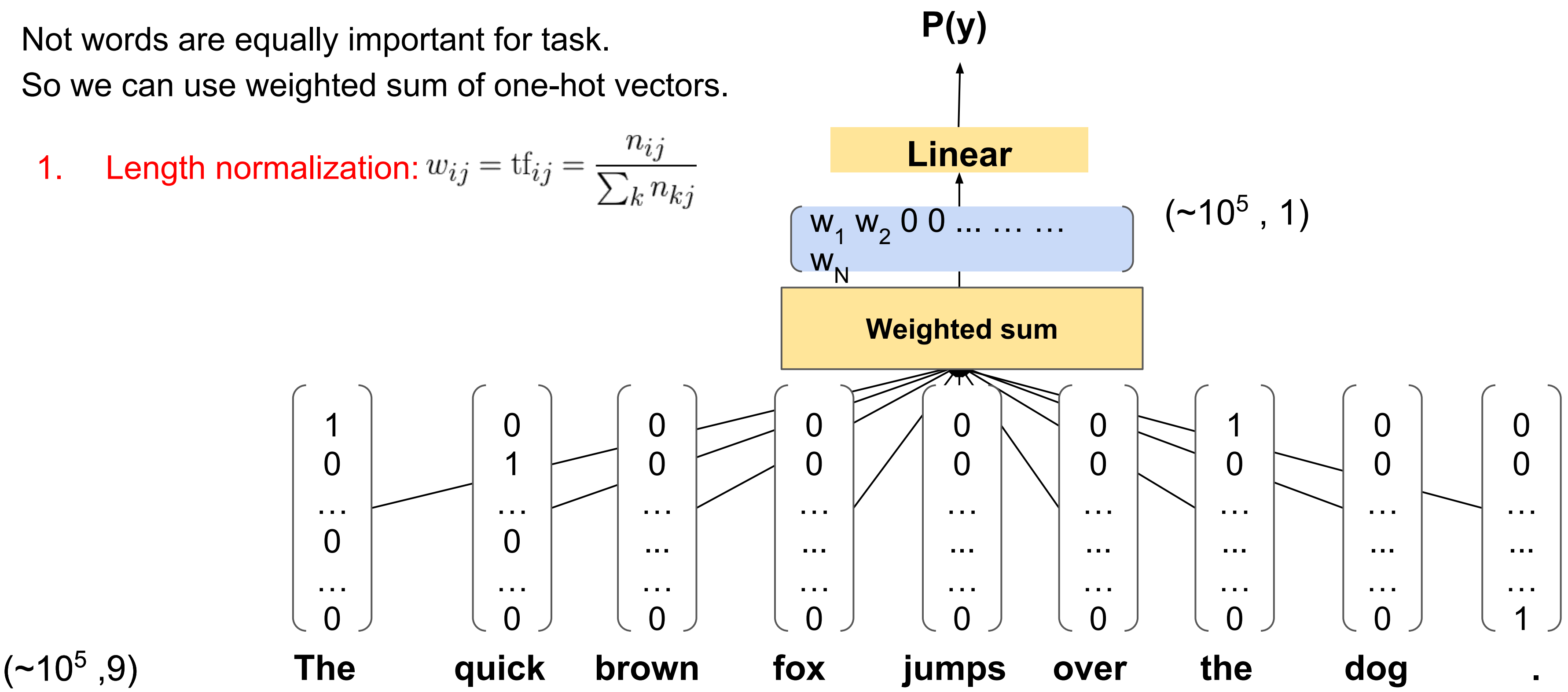
???



Weighting techniques for BOW

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

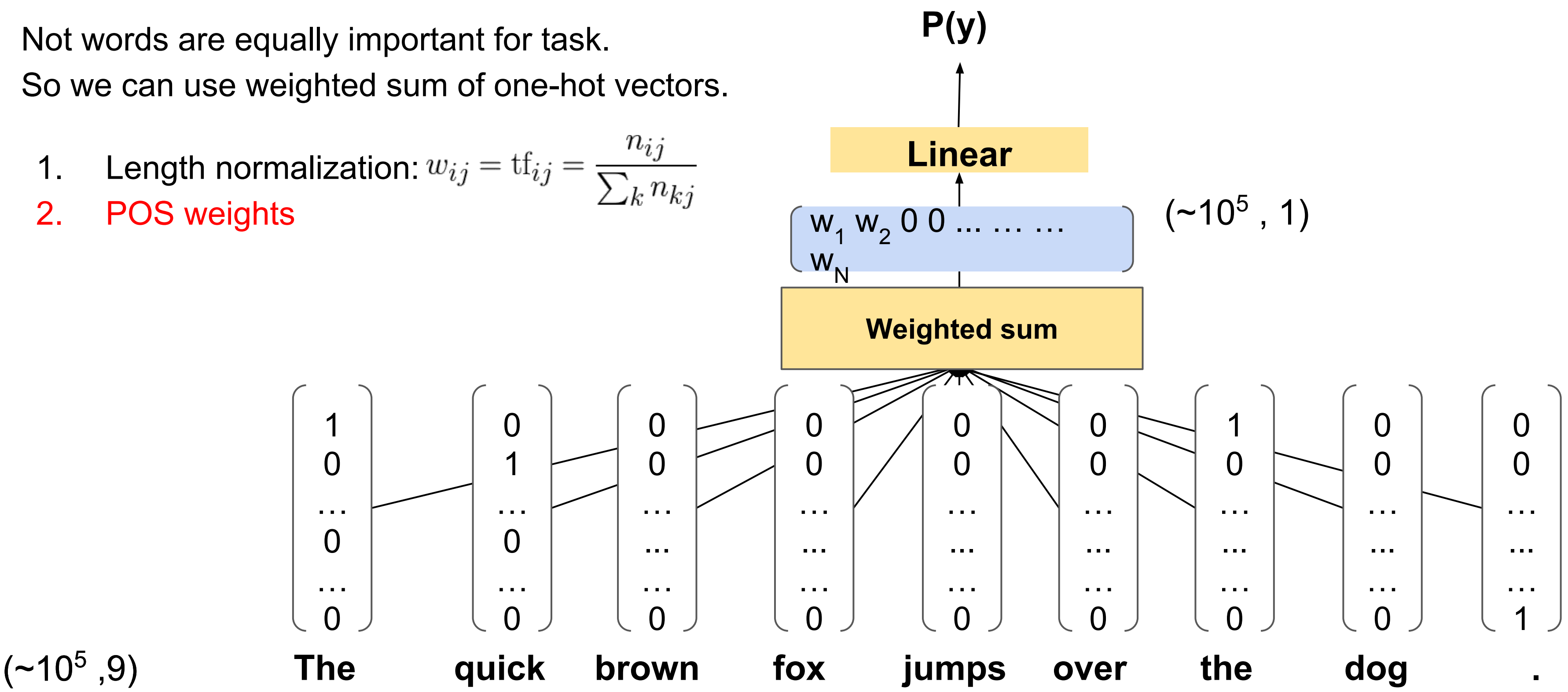
1. Length normalization: $w_{ij} = \mathrm{tf}_{ij} = \frac{n_{ij}}{\sum_k n_{kj}}$



Weighting techniques for BOW

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

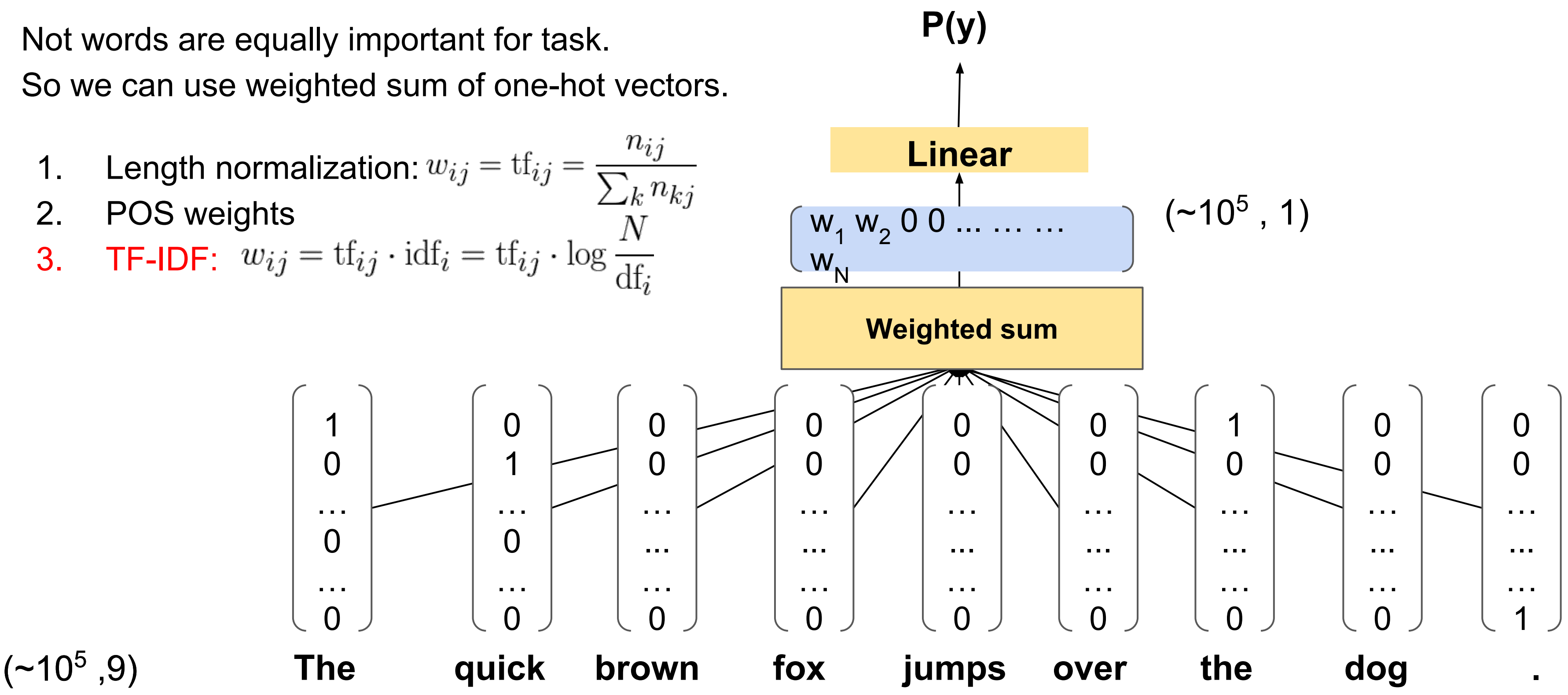
1. Length normalization: $w_{ij} = \mathrm{tf}_{ij} = \frac{n_{ij}}{\sum_k n_{kj}}$
2. POS weights



Weighting techniques for BOW

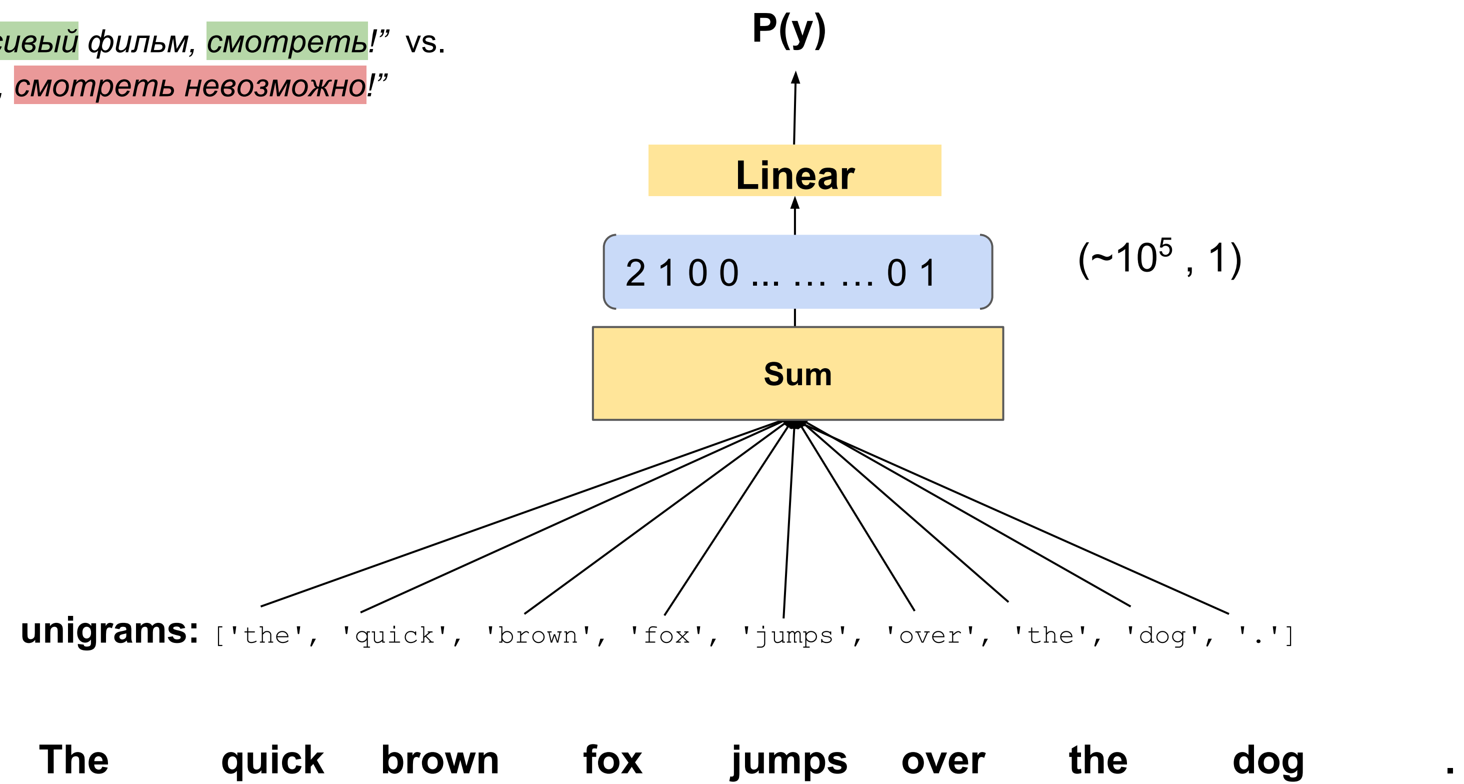
Not words are equally important for task.
So we can use weighted sum of one-hot vectors.

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

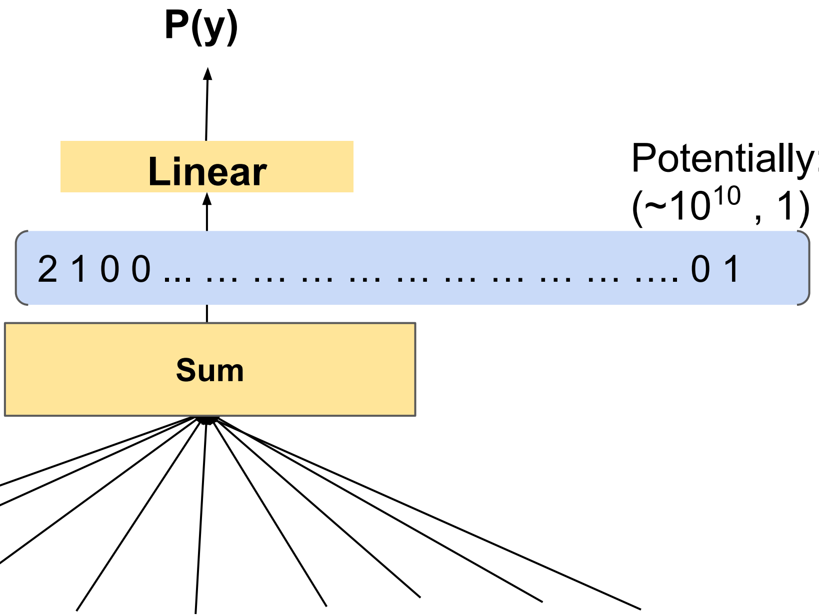
"Невозможно красивый фильм, смотреть!" vs.
"Красивый фильм, смотреть невозможно!"



Context importance

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

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



Potentially:
(~10^{10} , 1)

(~10^{10} ,17)

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**”.

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

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

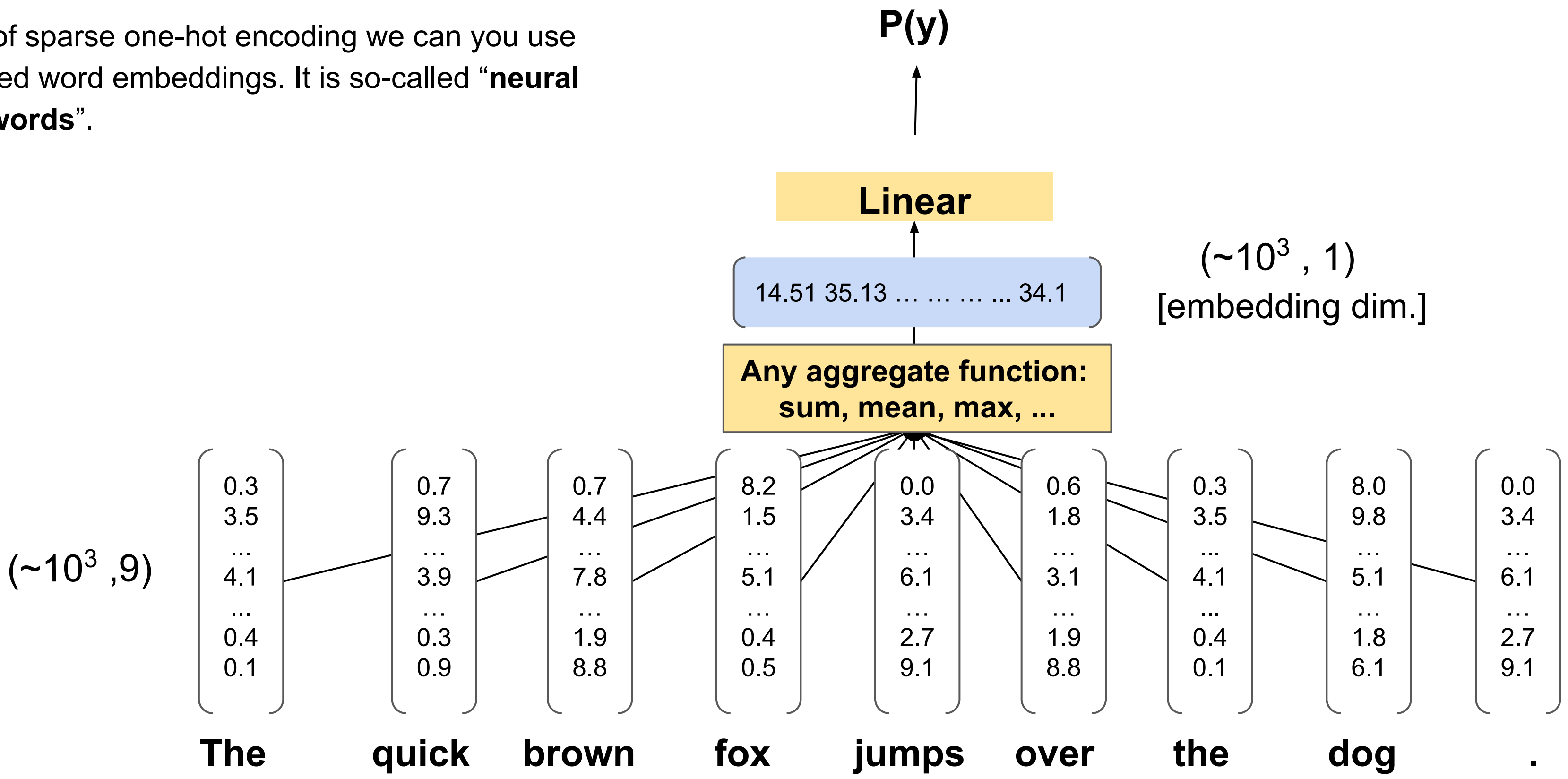
Dense text representation: NBOW

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

$$
(\sim 10^{3}, 9) \quad
\underset{\textbf{The}}{\begin{pmatrix} 0.3 \\ 3.5 \\ \dots \\ 4.1 \\ \dots \\ 0.4 \\ 0.1 \end{pmatrix}}
\underset{\textbf{quick}}{\begin{pmatrix} 0.7 \\ 9.3 \\ \dots \\ 3.9 \\ \dots \\ 0.3 \\ 0.9 \end{pmatrix}}
\underset{\textbf{brown}}{\begin{pmatrix} 0.7 \\ 4.4 \\ \dots \\ 7.8 \\ \dots \\ 1.9 \\ 8.8 \end{pmatrix}}
\underset{\textbf{fox}}{\begin{pmatrix} 8.2 \\ 1.5 \\ \dots \\ 5.1 \\ \dots \\ 0.4 \\ 0.5 \end{pmatrix}}
\underset{\textbf{jumps}}{\begin{pmatrix} 0.0 \\ 3.4 \\ \dots \\ 6.1 \\ \dots \\ 2.7 \\ 9.1 \end{pmatrix}}
\underset{\textbf{over}}{\begin{pmatrix} 0.6 \\ 1.8 \\ \dots \\ 3.1 \\ \dots \\ 1.9 \\ 8.8 \end{pmatrix}}
\underset{\textbf{the}}{\begin{pmatrix} 0.3 \\ 3.5 \\ \dots \\ 4.1 \\ \dots \\ 0.4 \\ 0.1 \end{pmatrix}}
\underset{\textbf{dog}}{\begin{pmatrix} 8.0 \\ 9.8 \\ \dots \\ 5.1 \\ \dots \\ 1.8 \\ 6.1 \end{pmatrix}}
\underset{\textbf{.}}{\begin{pmatrix} 0.0 \\ 3.4 \\ \dots \\ 6.1 \\ \dots \\ 2.7 \\ 9.1 \end{pmatrix}}
$$

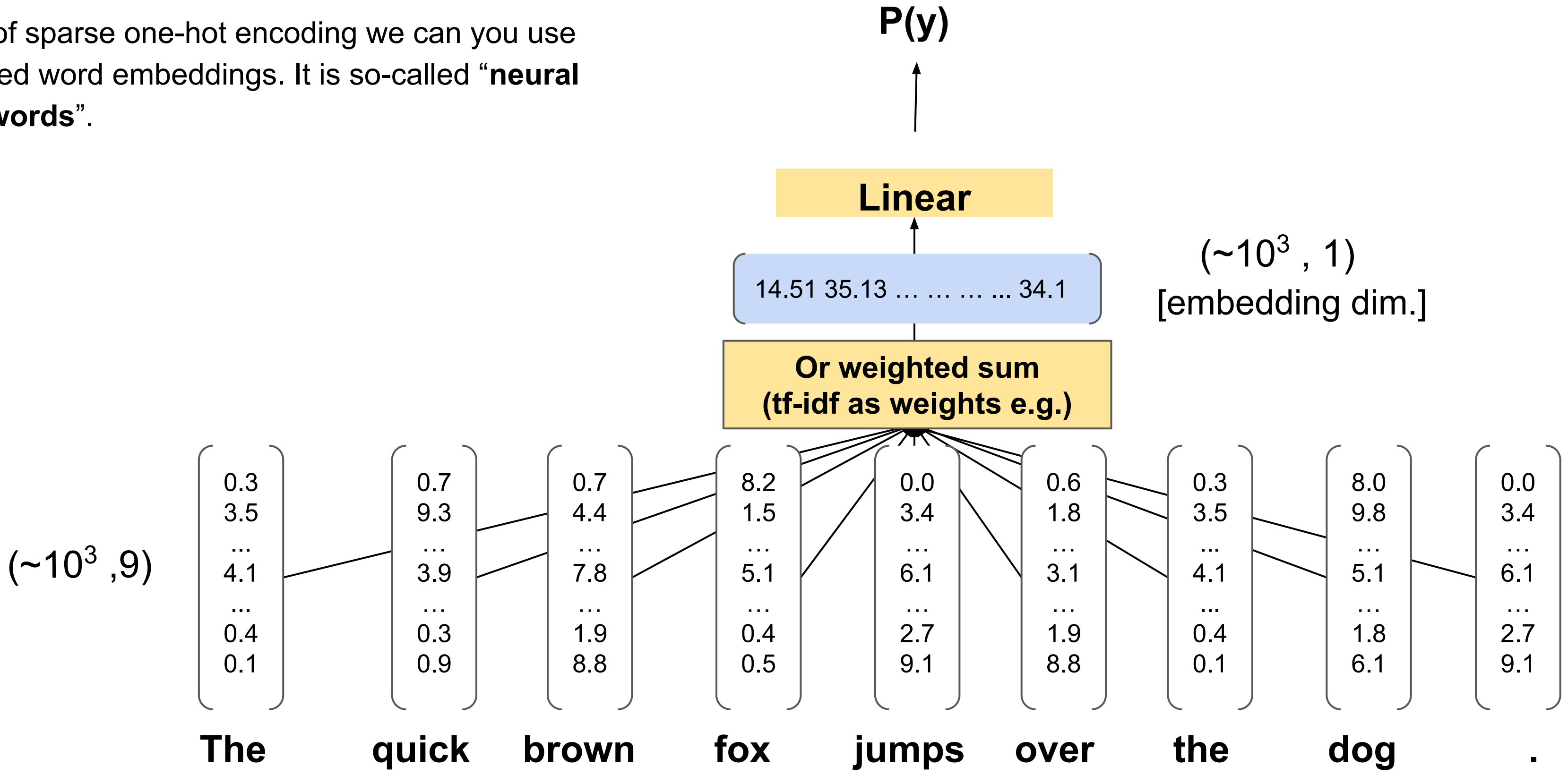
Dense text representation: NBOW

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



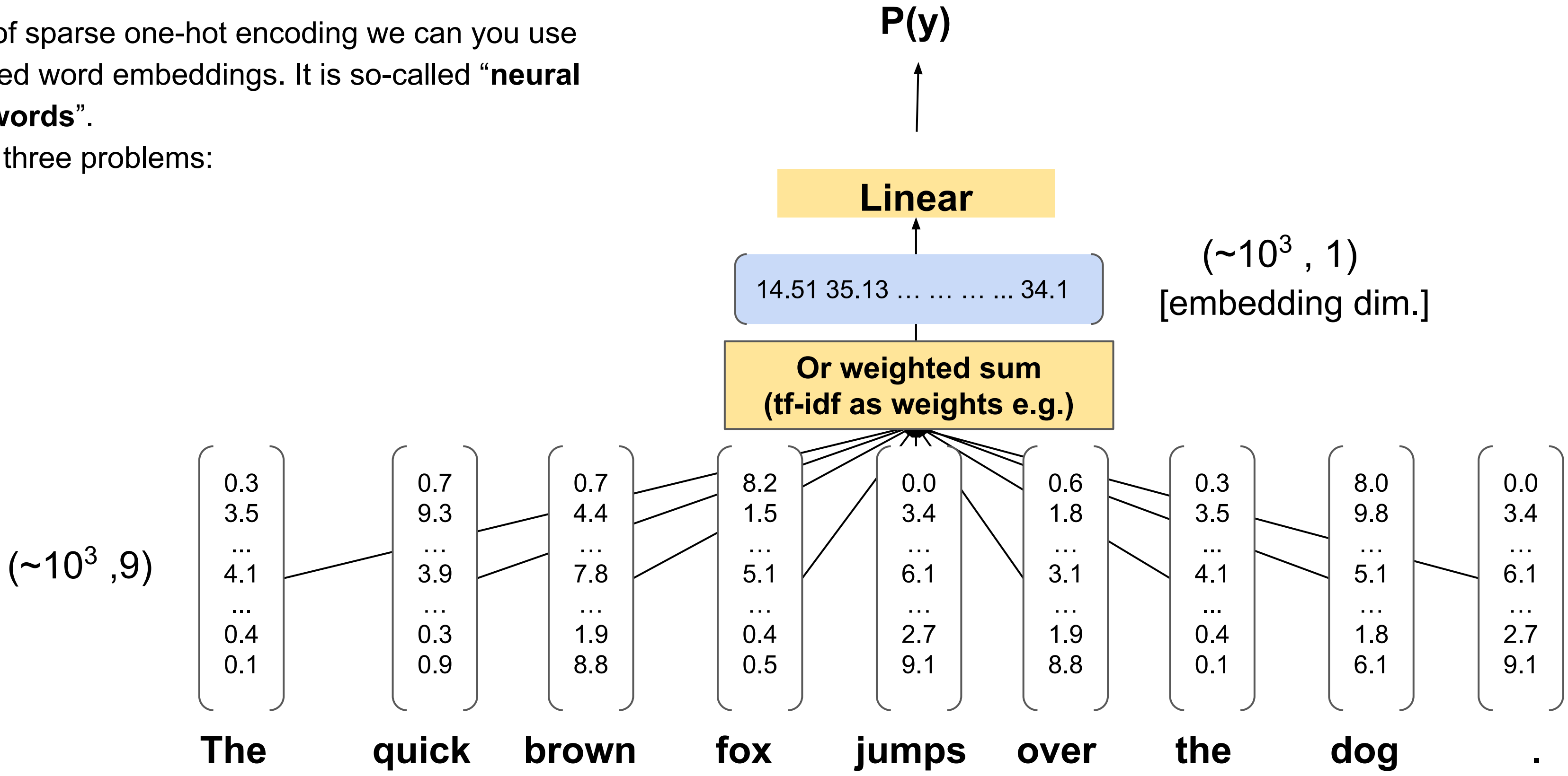
Dense text representation: NBOW

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

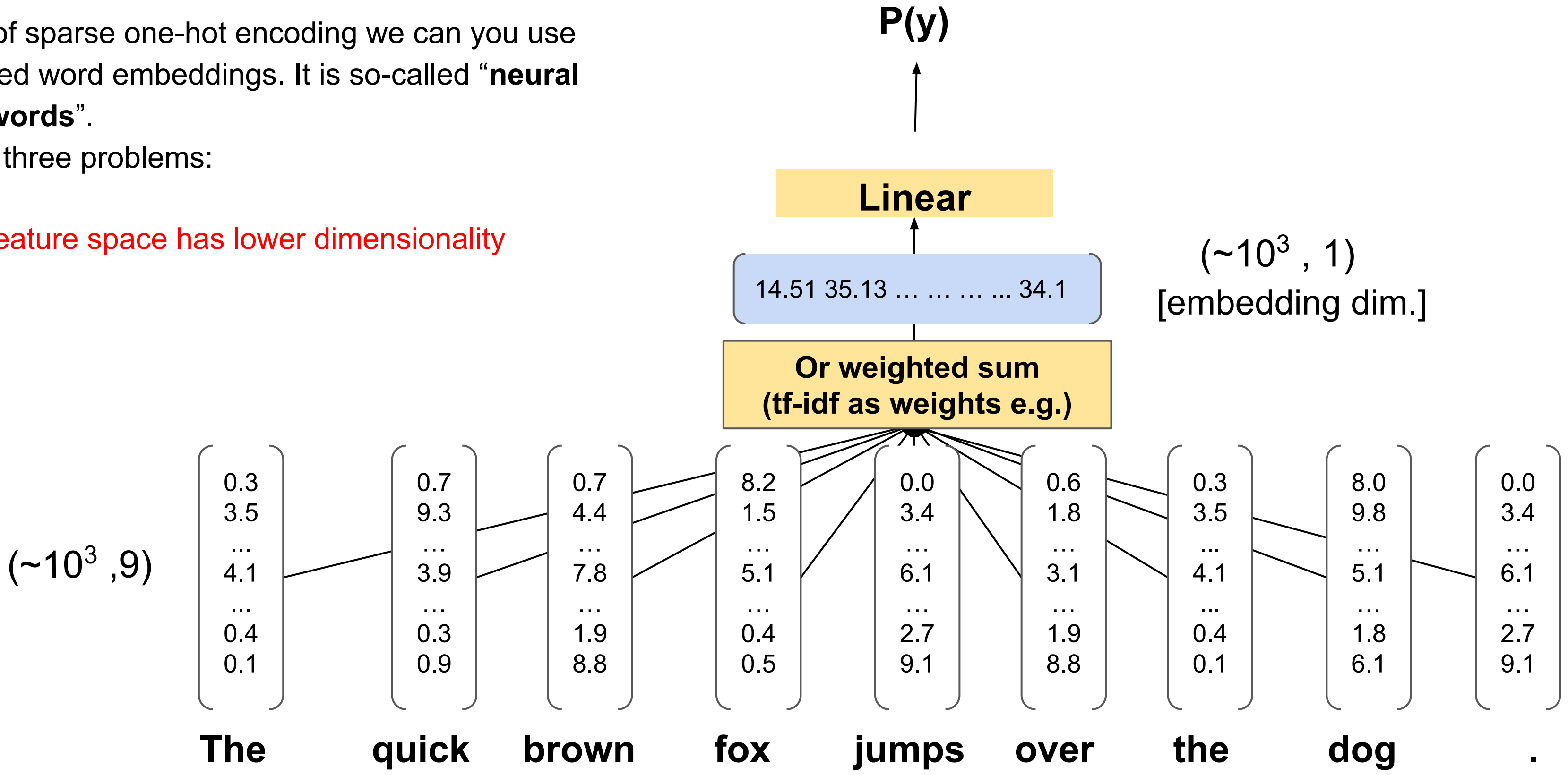
Instead of sparse one-hot encoding we can you use pre-trained word embeddings. It is so-called “**neural bag-of-words**”.
It solves three problems:



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:

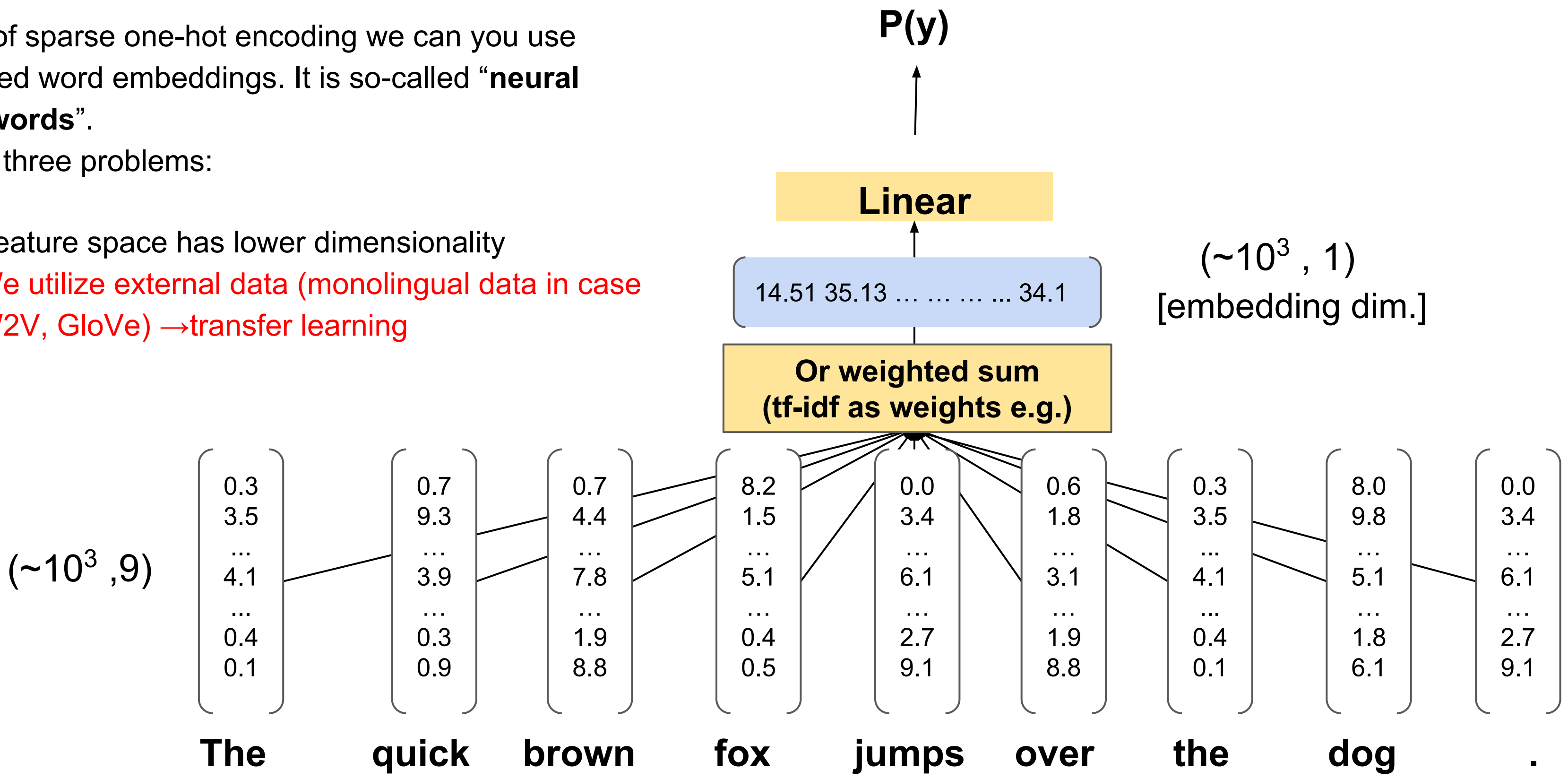
1. Feature space has lower dimensionality



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:

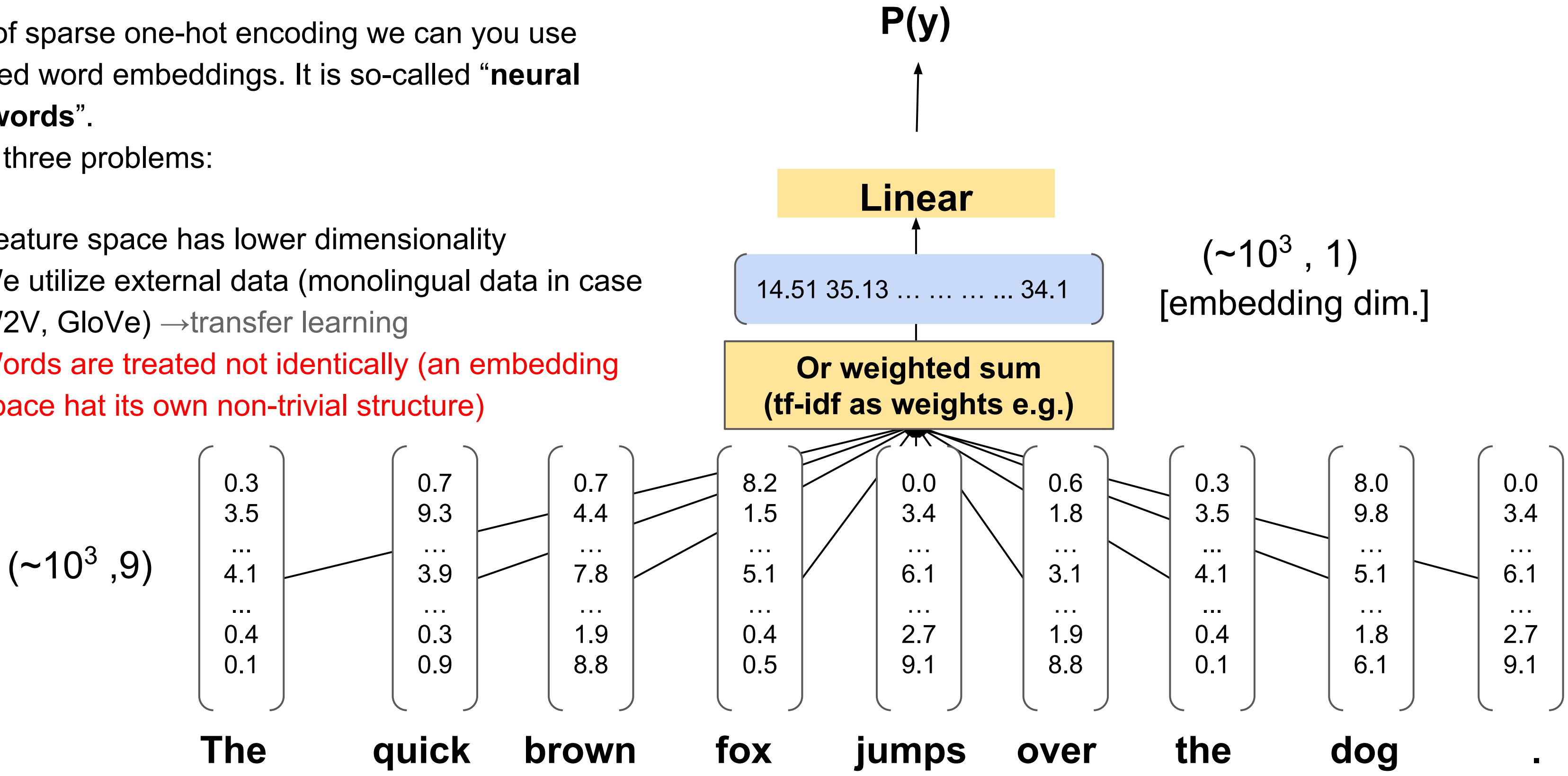
1. Feature space has lower dimensionality
2. We utilize external data (monolingual data in case W2V, GloVe) →transfer learning



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 hat its own non-trivial structure)



Dense text representation: NBOW

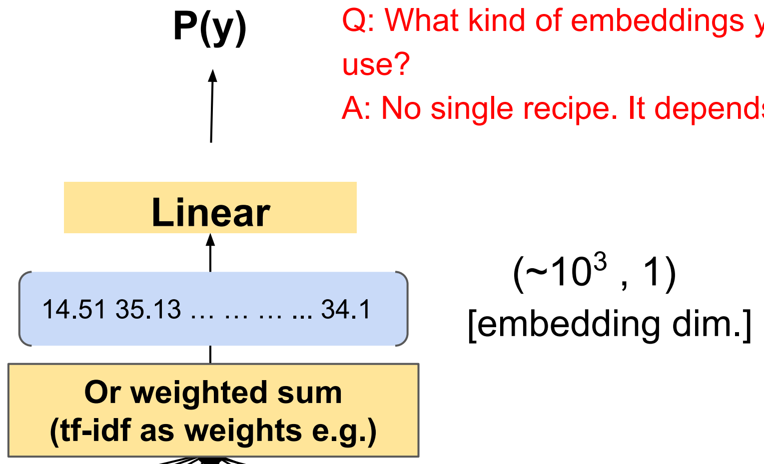
Instead of sparse one-hot encoding we can you use pre-trained word embeddings. It is so-called “**neural bag-of-words**”.

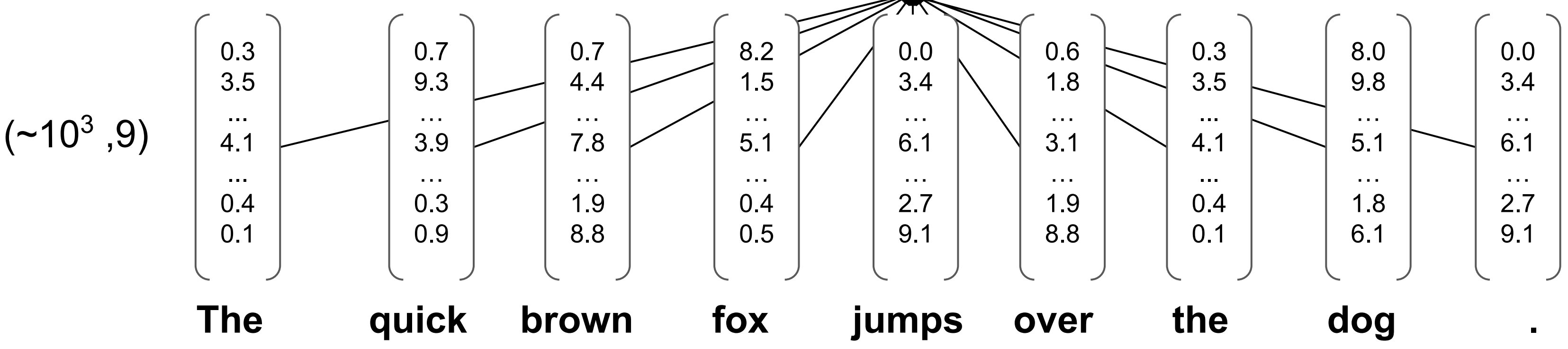
It solves three problems:

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 hat its own non-trivial structure)

Q: What kind of embeddings you should use?

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



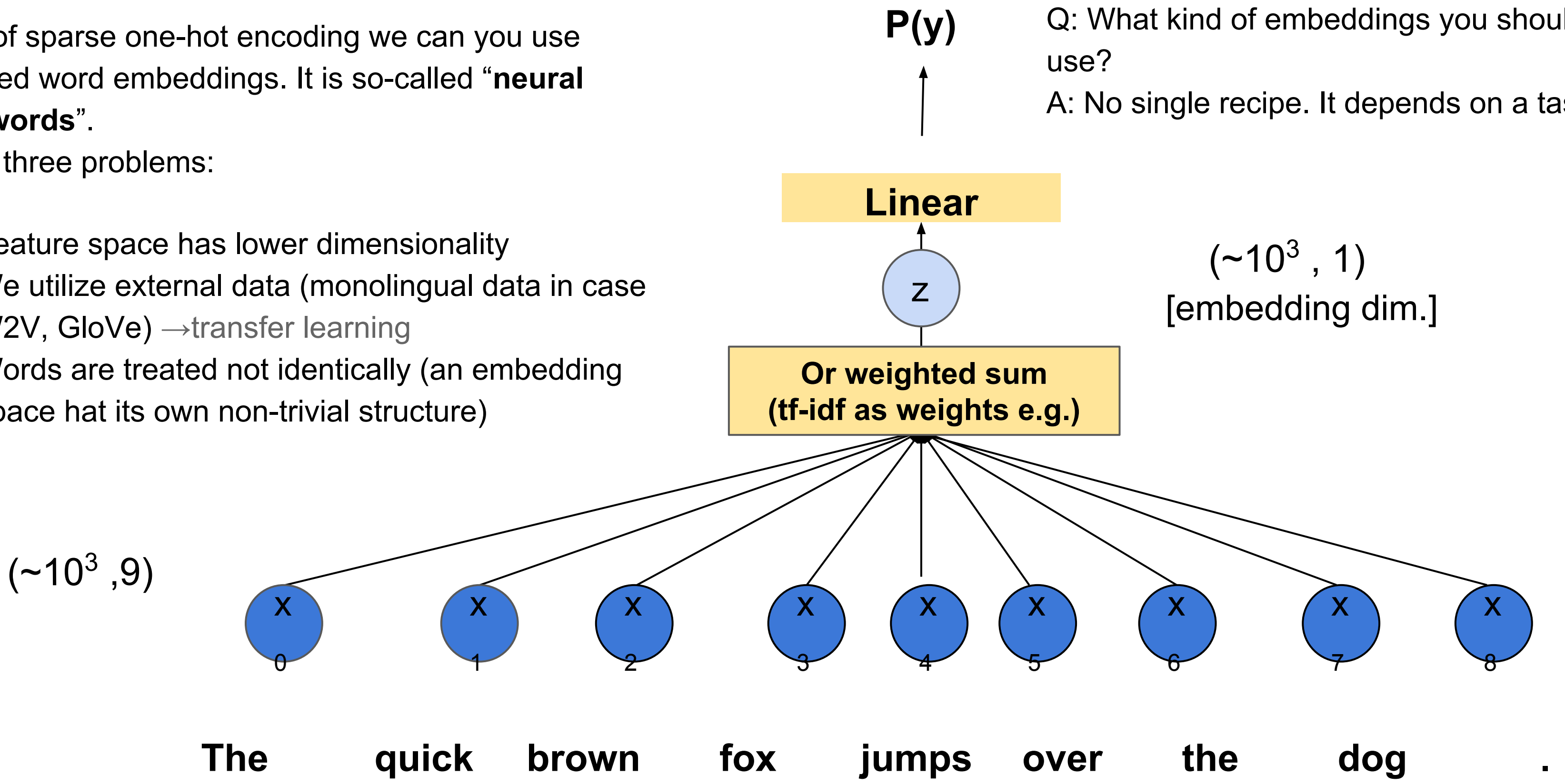


Dense text representation: NBOW

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...

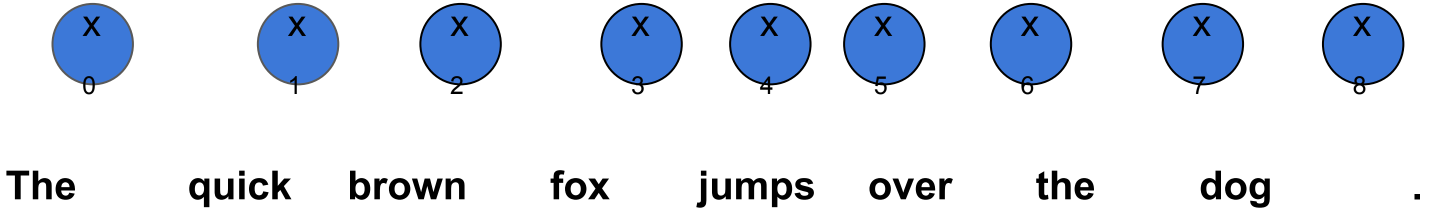
BOW and NBOW: the shared problems

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)

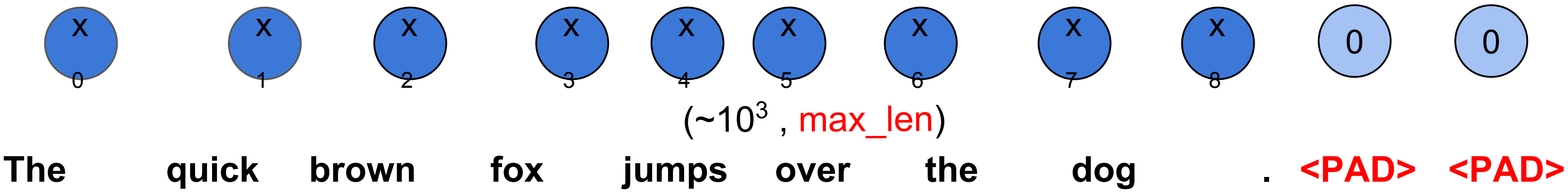
Multilayer perceptron

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



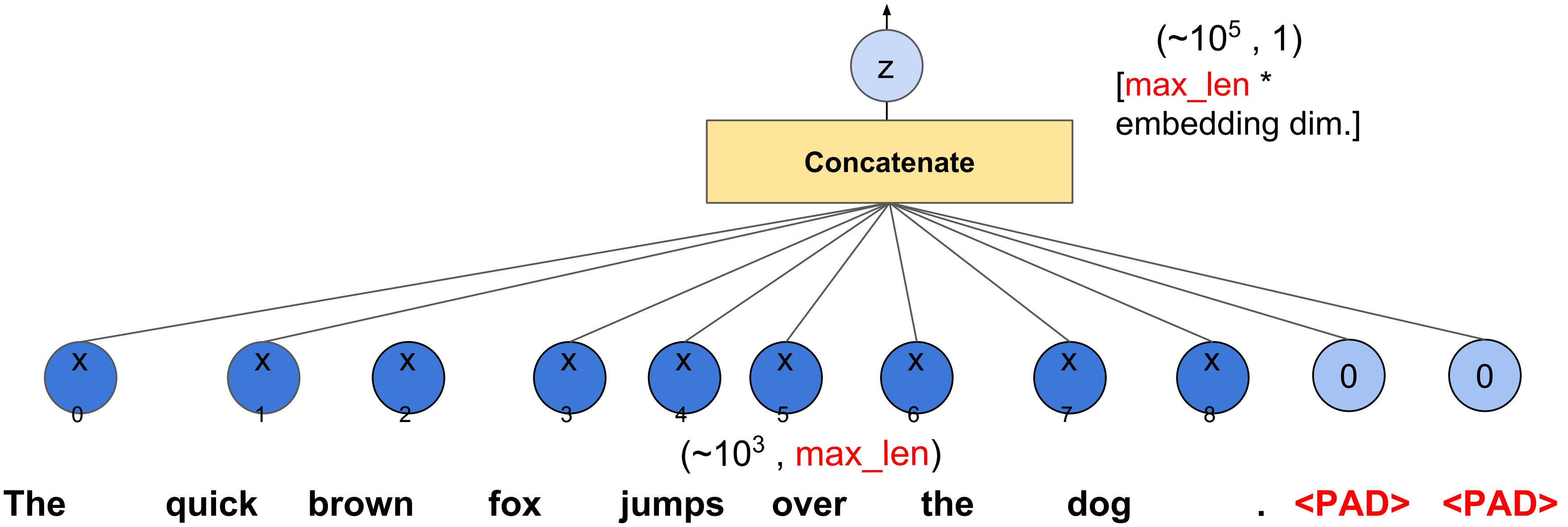
Multilayer perceptron

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



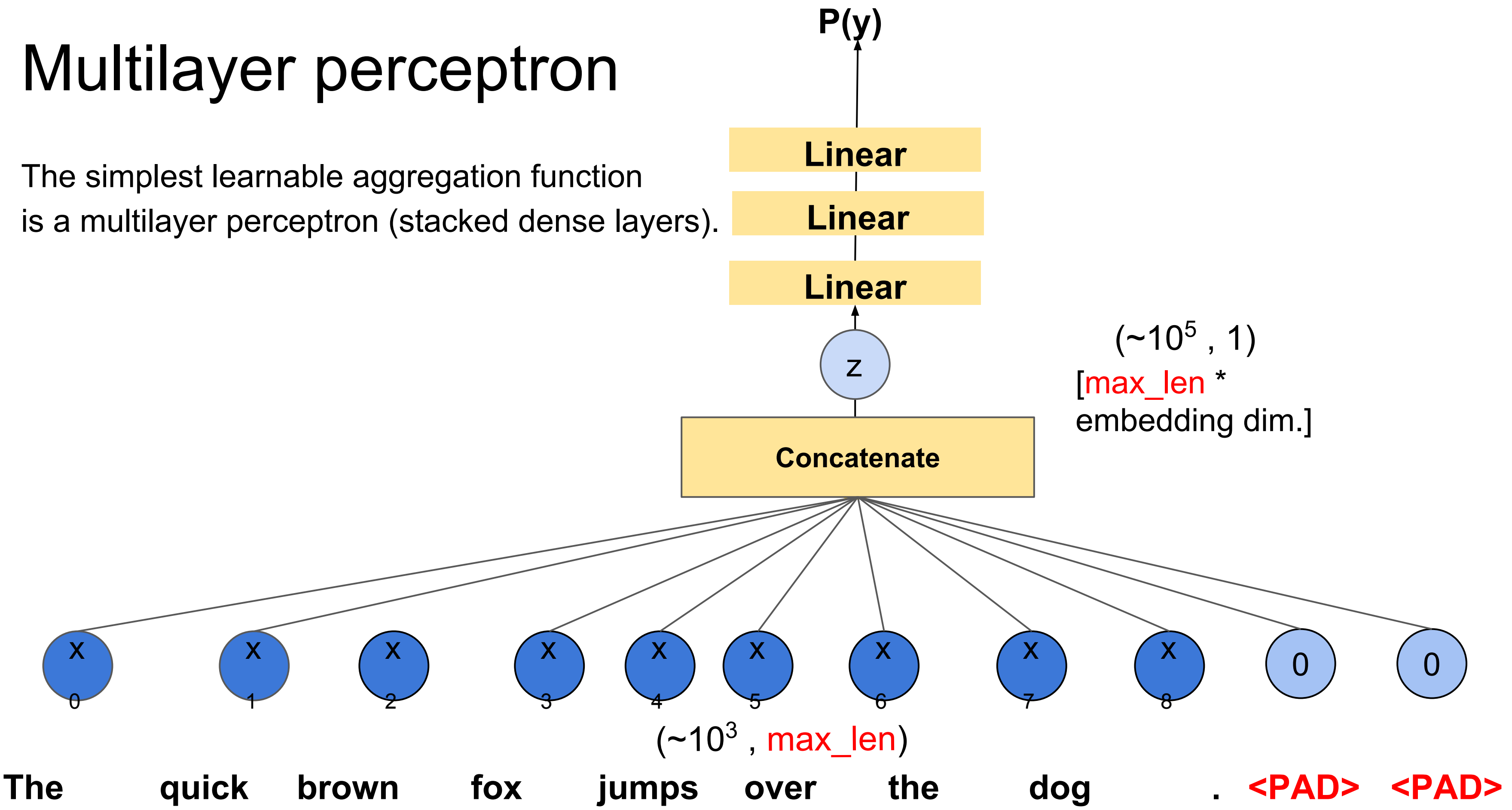
Multilayer perceptron

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).

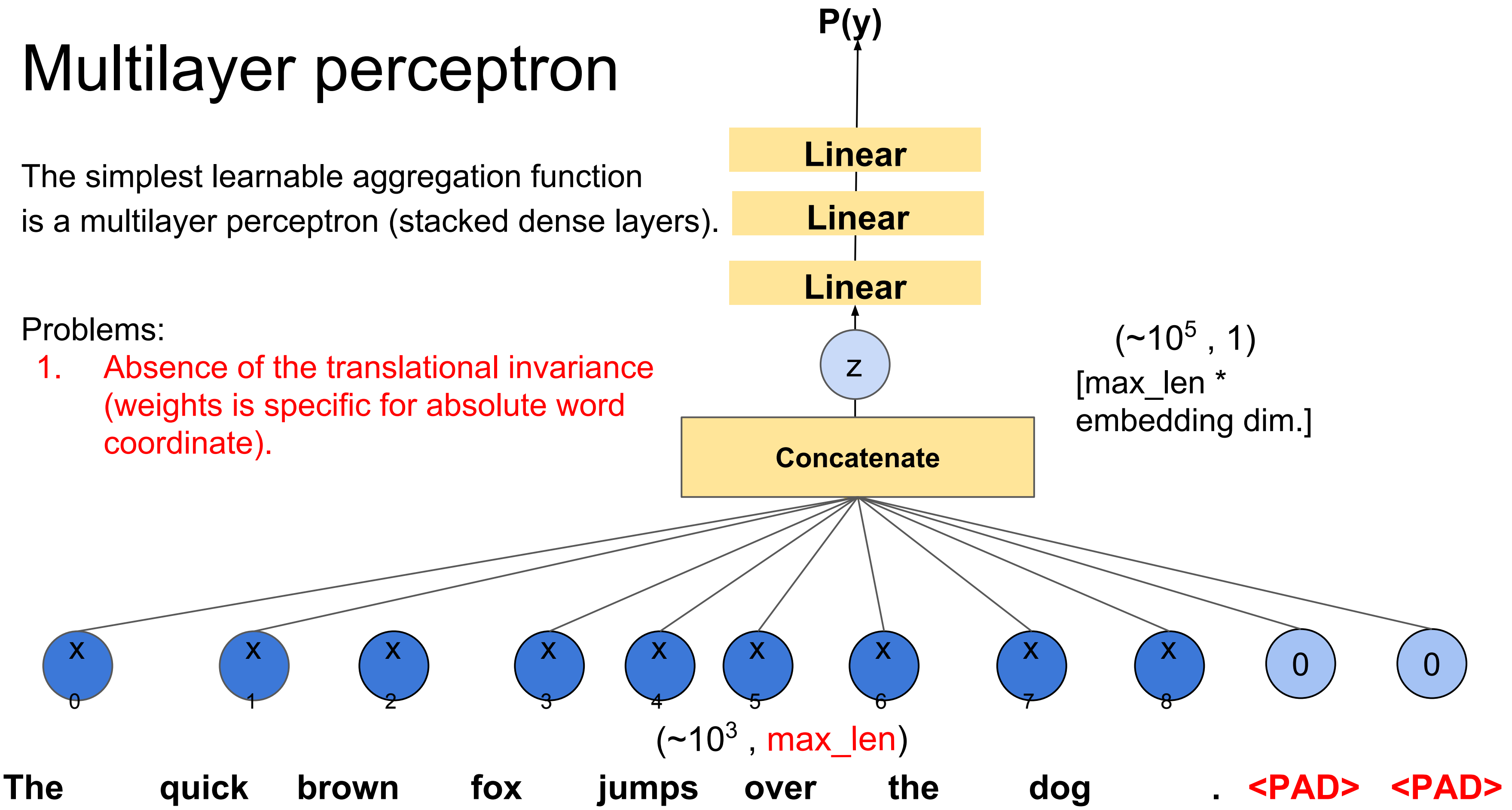


Multilayer perceptron

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).

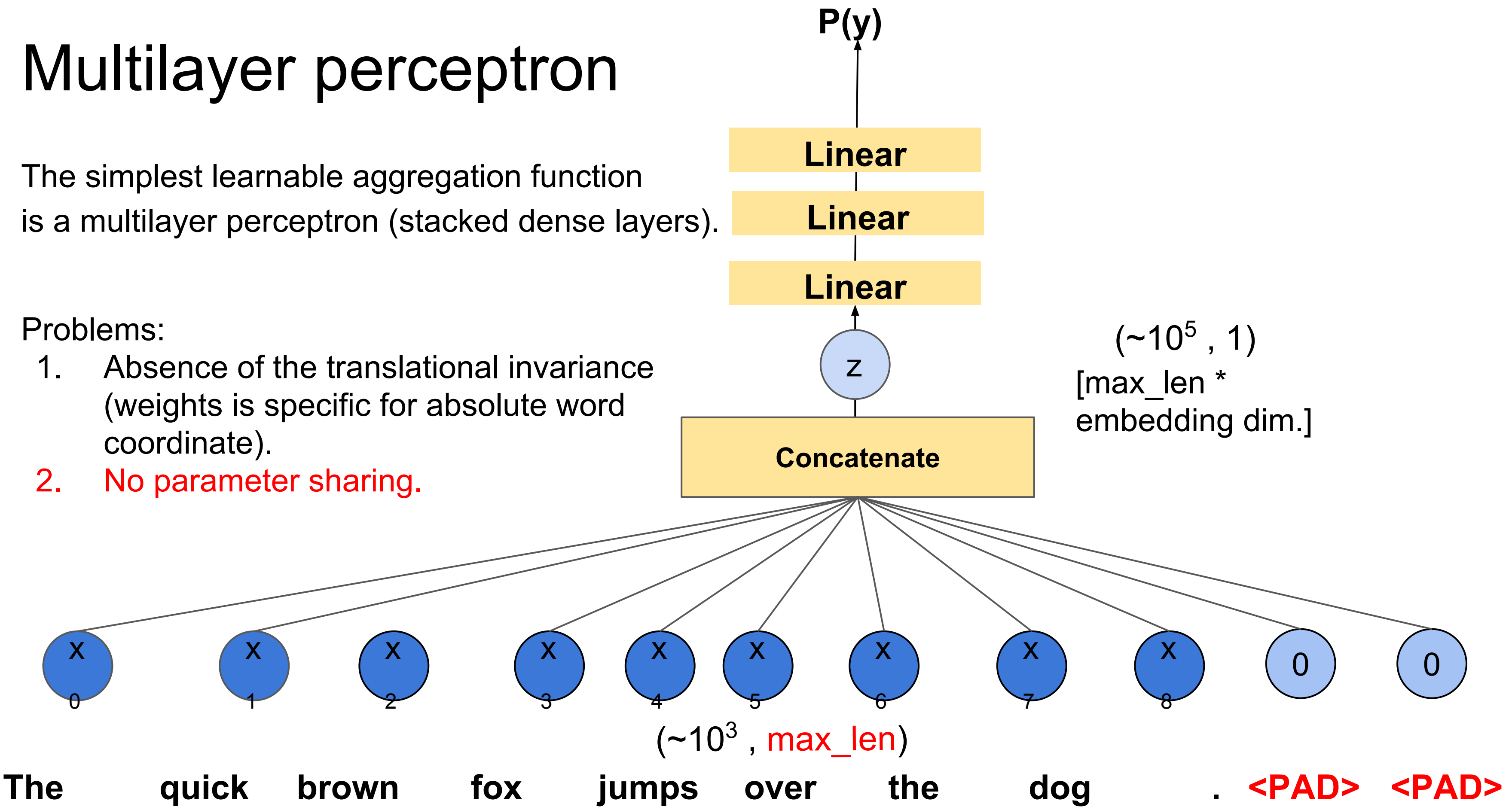


Multilayer perceptron

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.

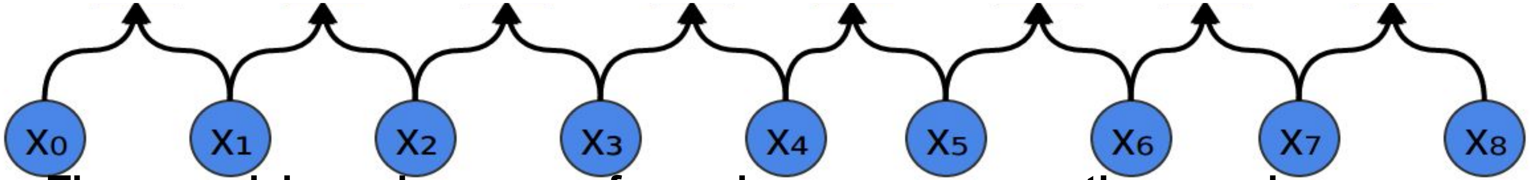


CNN for texts

(100,9)

CNN for texts

(100,9)



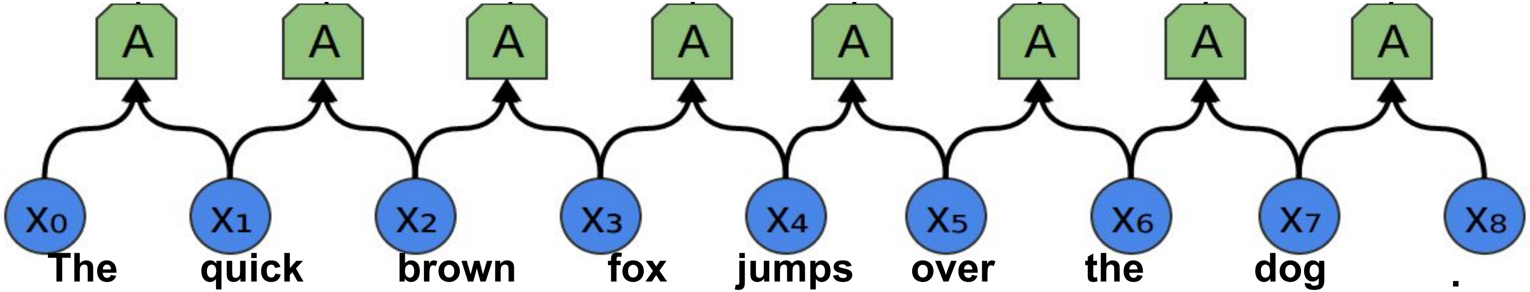
The quick brown fox jumps over the dog .

CNN for texts

A convolution kernel is a tensor of size
[output dim, embedding dim, kernel size]

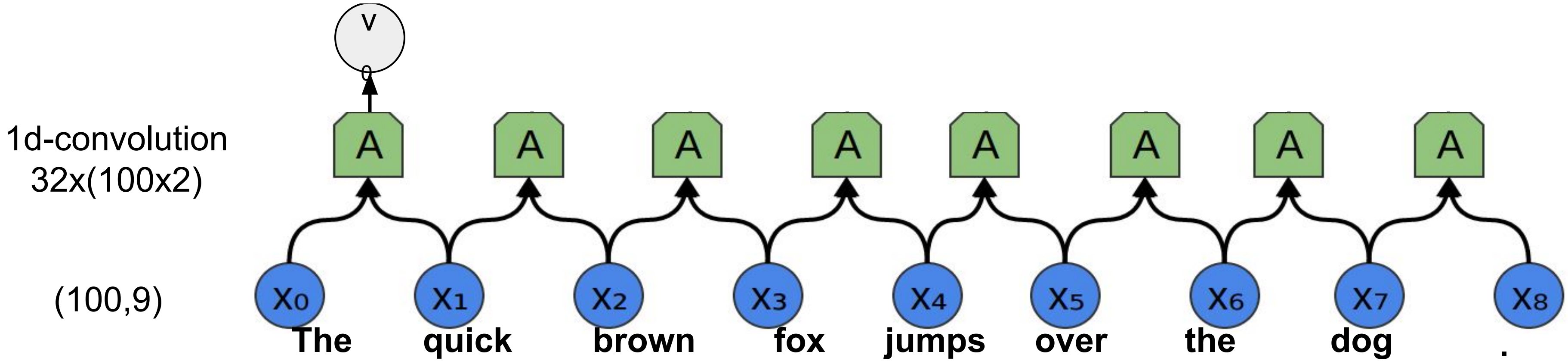
1d-convolution
32x(100x2)

(100,9)



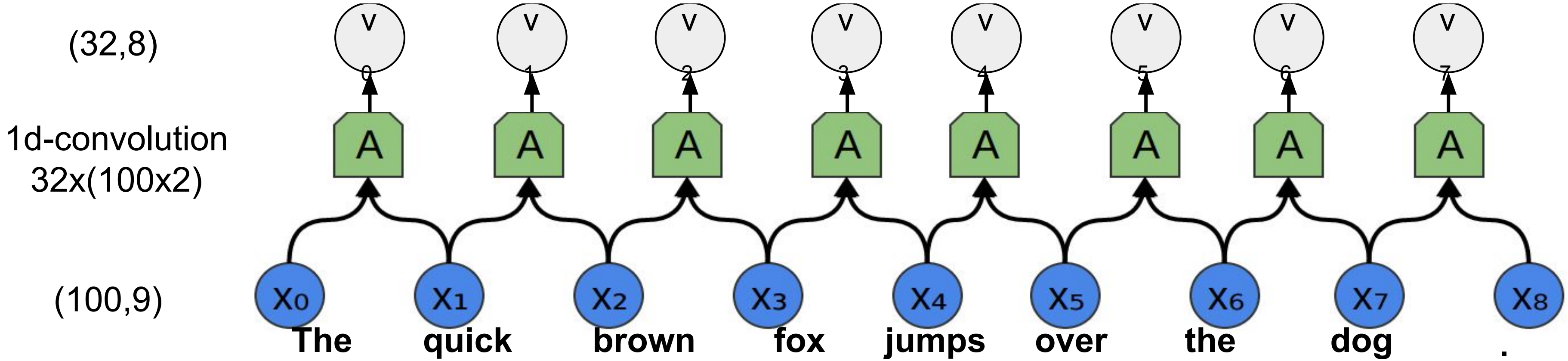
CNN for texts

$$\mathbf{v}_0 = \mathbf{A}(\mathbf{x}_0, \mathbf{x}_1)$$
$$v_{0i} = A_i(\mathbf{x}_0, \mathbf{x}_1) = \sum_j (K_{0ij}x_{0j} + K_{1ij}x_{1j})$$



CNN for texts

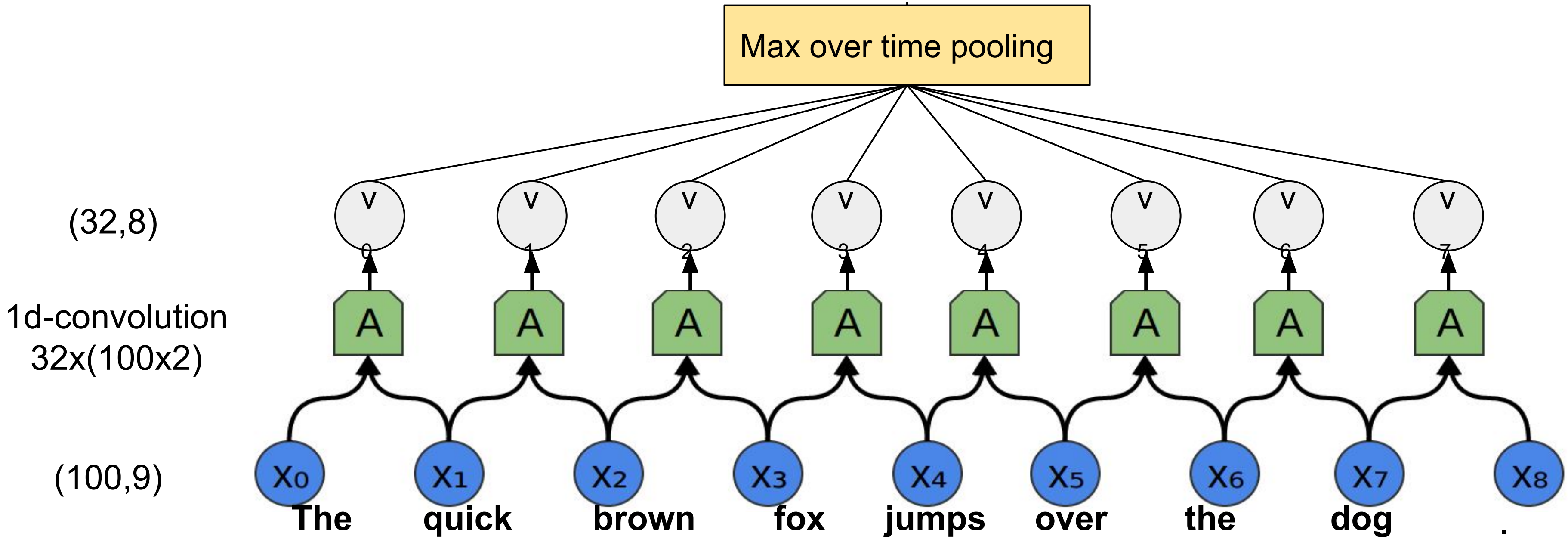
$$\mathbf{v}_0 = \mathbf{A}(\mathbf{x}_0, \mathbf{x}_1)$$
$$v_{0i} = A_i(\mathbf{x}_0, \mathbf{x}_1) = \sum_j (K_{0ij} x_{0j} + K_{1ij} x_{1j})$$



CNN for texts

$$\mathbf{v}_0 = \mathbf{A}(\mathbf{x}_0, \mathbf{x}_1)$$

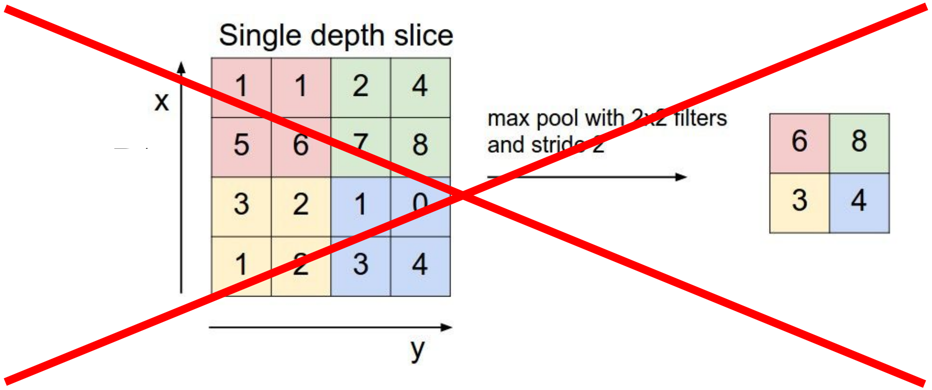
$$v_{0i} = A_i(\mathbf{x}_0, \mathbf{x}_1) = \sum_j (K_{0ij}x_{0j} + K_{1ij}x_{1j})$$

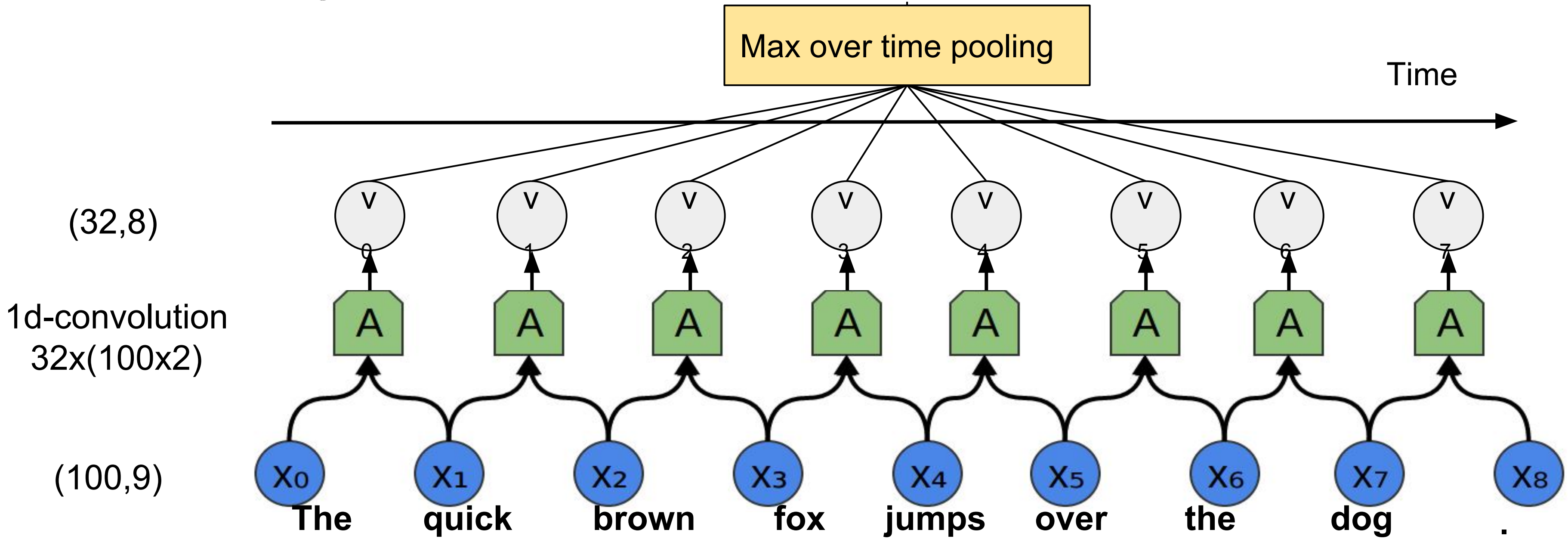


CNN for texts

$$\mathbf{v}_0 = \mathbf{A}(\mathbf{x}_0, \mathbf{x}_1)$$

$$v_{0i} = A_i(\mathbf{x}_0, \mathbf{x}_1) = \sum_j (K_{0ij}x_{0j} + K_{1ij}x_{1j})$$

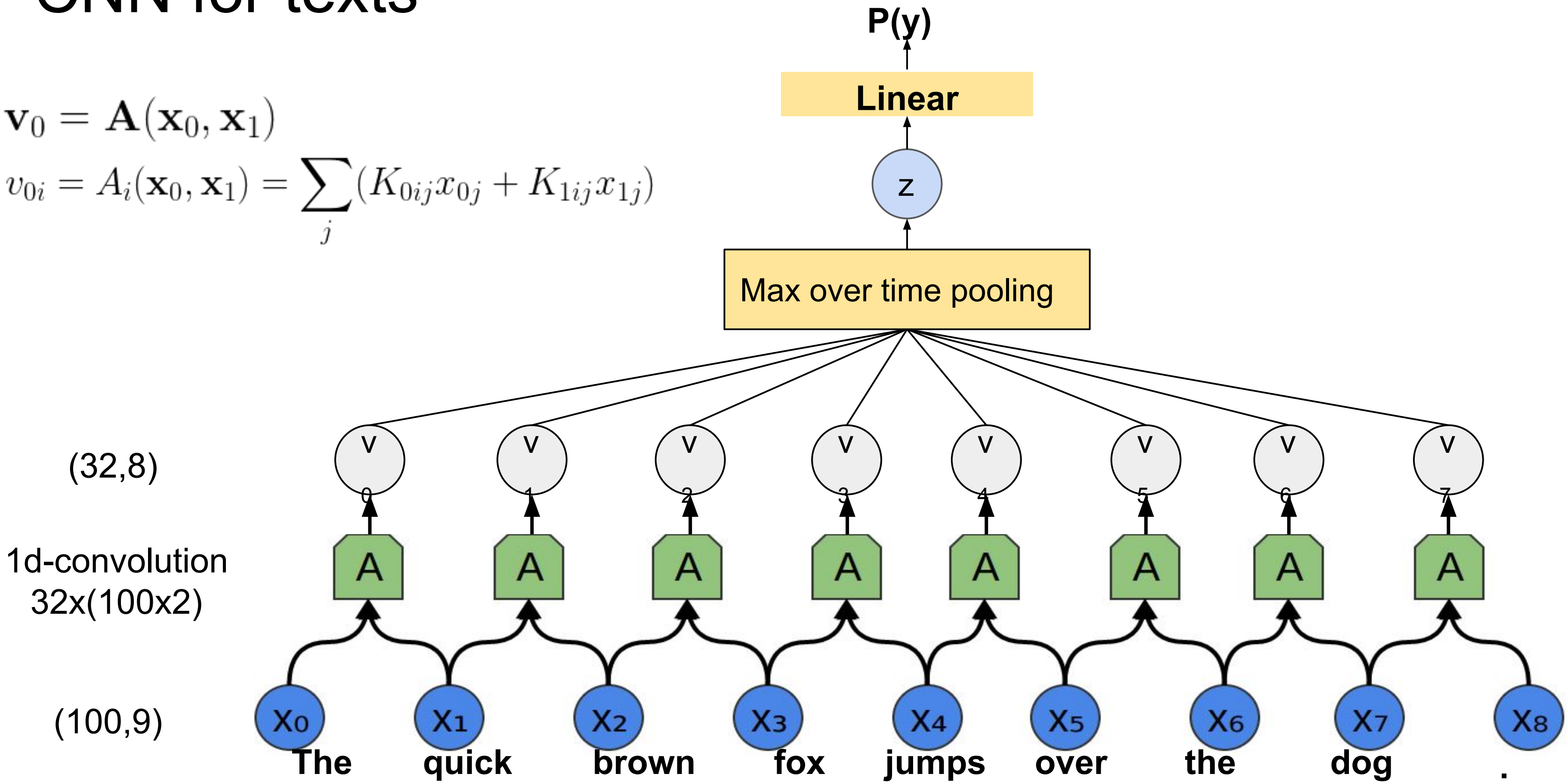




CNN for texts

$$\mathbf{v}_0 = \mathbf{A}(\mathbf{x}_0, \mathbf{x}_1)$$

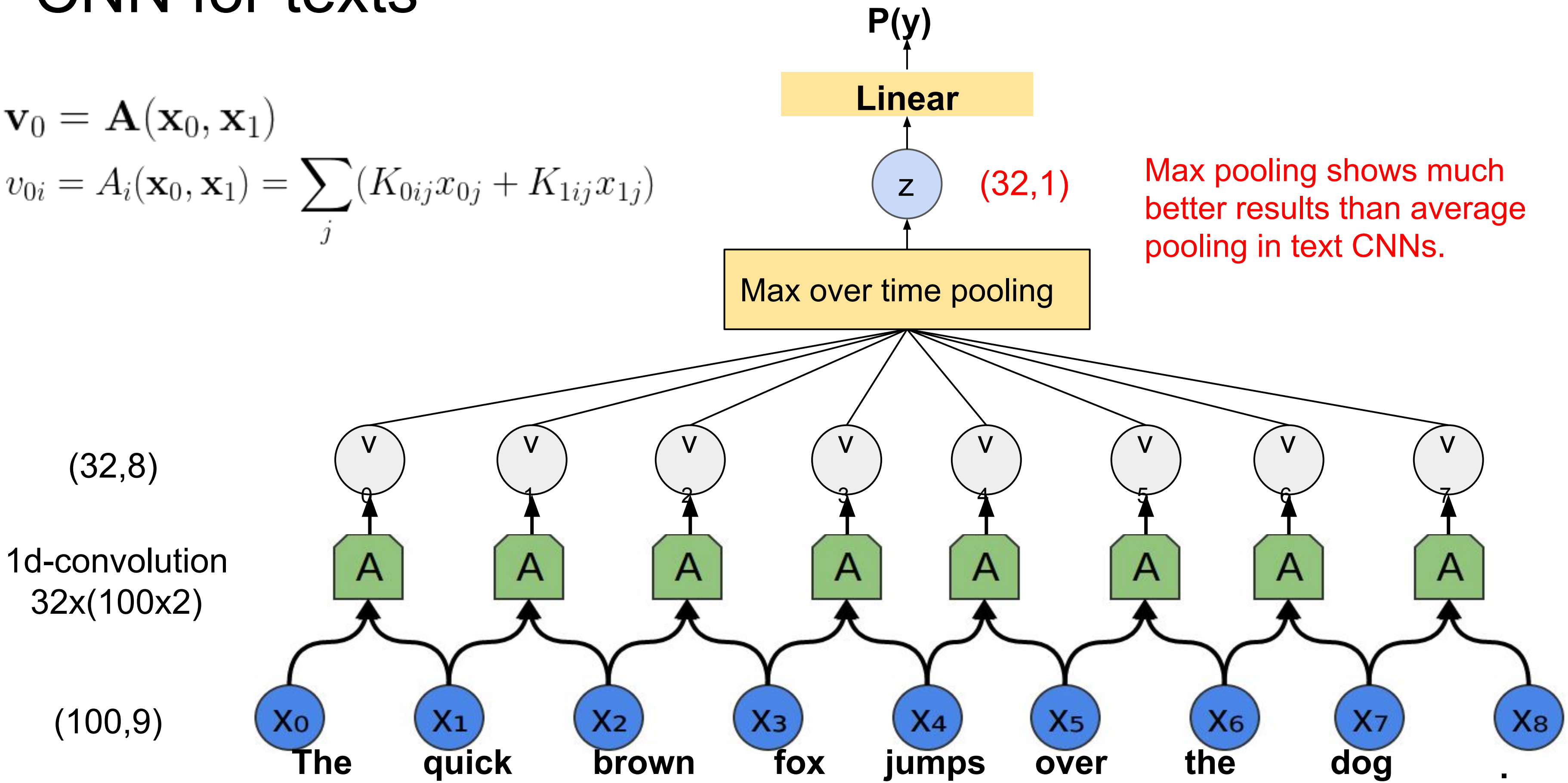
$$v_{0i} = A_i(\mathbf{x}_0, \mathbf{x}_1) = \sum_j (K_{0ij} x_{0j} + K_{1ij} x_{1j})$$



CNN for texts

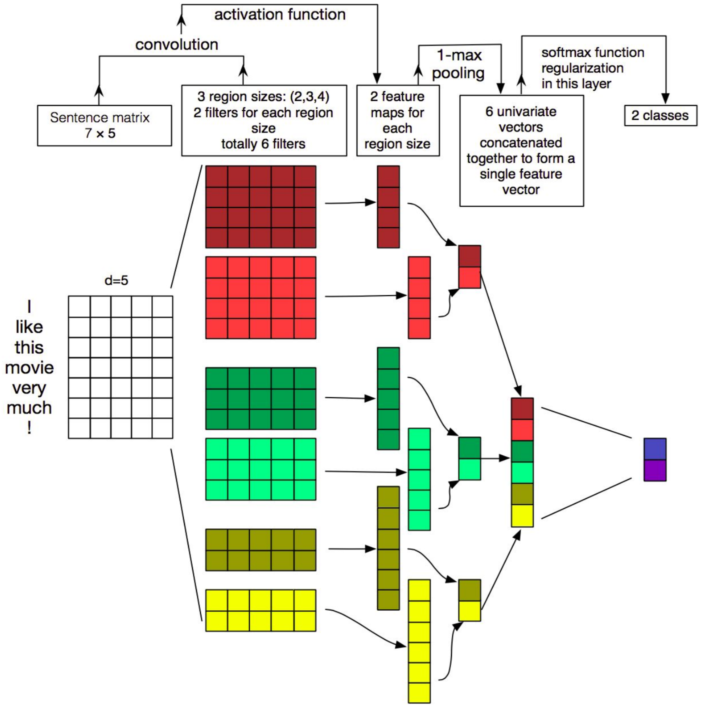
$$\mathbf{v}_0 = \mathbf{A}(\mathbf{x}_0, \mathbf{x}_1)$$

$$v_{0i} = A_i(\mathbf{x}_0, \mathbf{x}_1) = \sum_j (K_{0ij} x_{0j} + K_{1ij} x_{1j})$$



Max pooling shows much better results than average pooling in text CNNs.

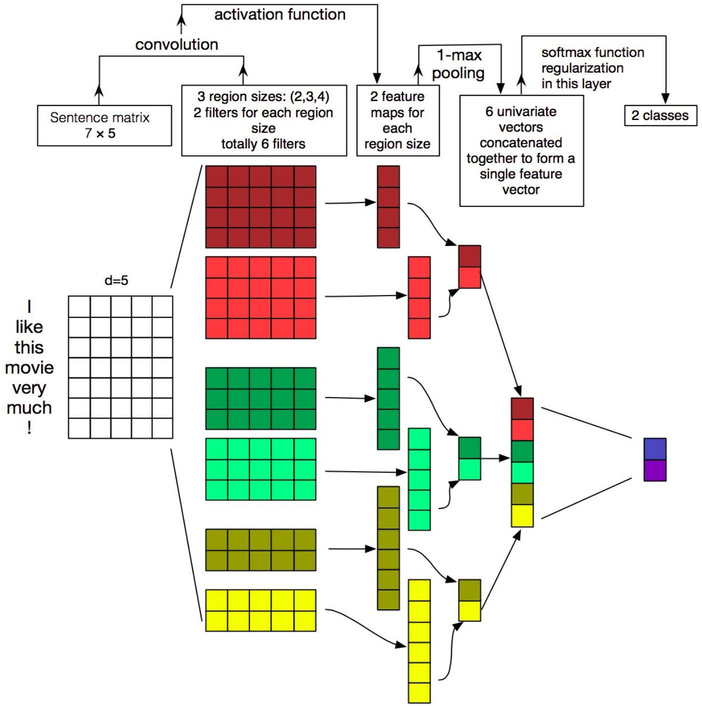
CNN for texts: improvements



- Use convolutional layers with different kernel size, separate max-pooling over time and concatenation.

[Zhang et al. 2015]

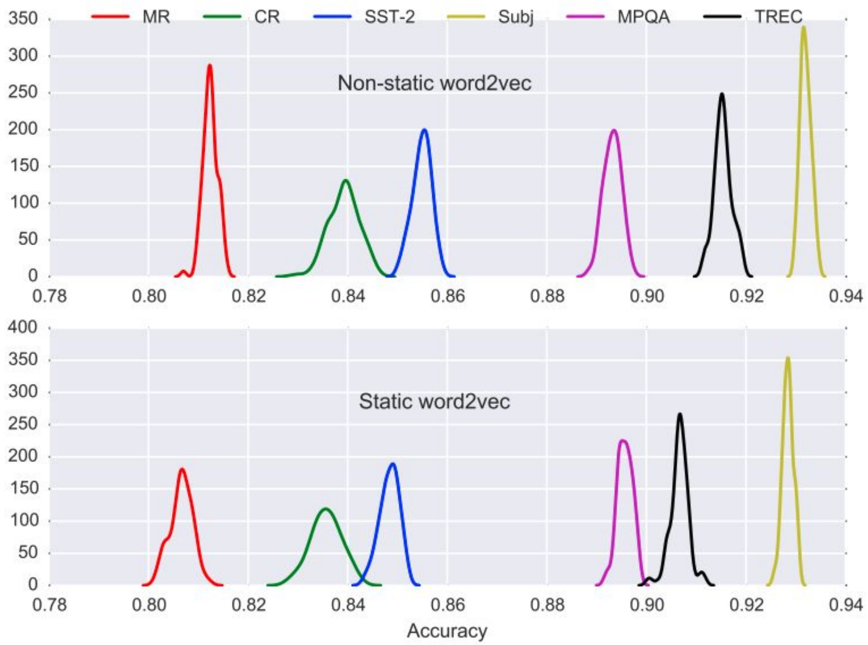
CNN for texts: improvements



- Use convolutional layers with different kernel size, separate max-pooling over time and concatenation.
- K-max pooling: take not 1 but k highest activations in their original order.
  E.g. (0,1,3,2,0,1,4,1) → (3,2,4)

[Zhang et al. 2015]

CNN for texts: improvements



Accuracy density plots for non-static w2v (upper) and static w2v (lower) [for 10-fold CV over the 100 replications]

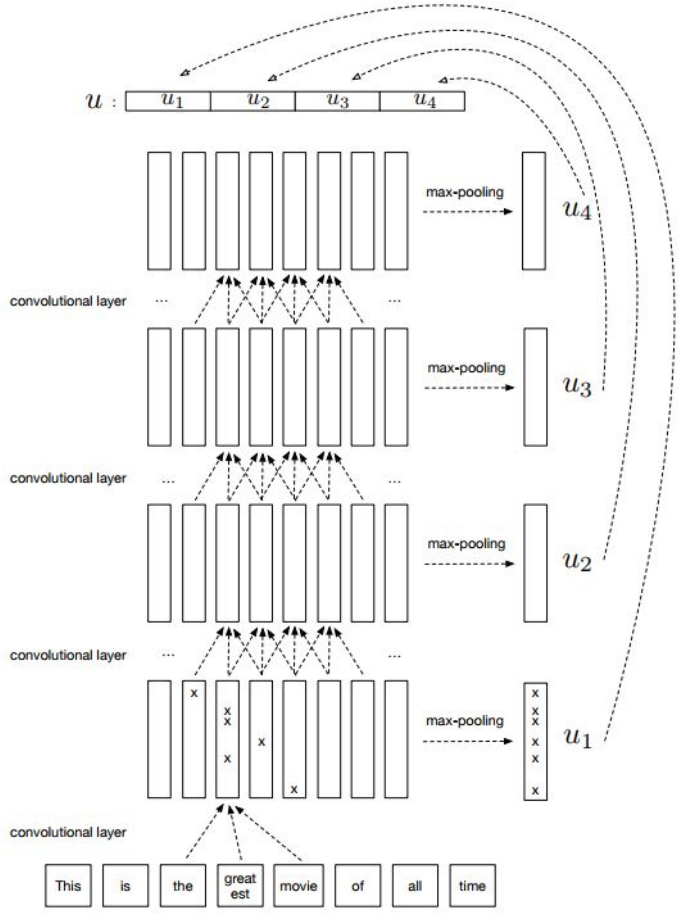
- Use convolutional layers with different kernel size, separate max-pooling over time and concatenation.
- K-max pooling: take not 1 but k highest activations in their original order.
  E.g. (0,1,3,2,0,1,4,1) → (3,2,4)
- Use pre-trained word vectors only for embedding layer initialization, train it jointly with model

[Zhang et al. 2015]

Should we stack more layers?

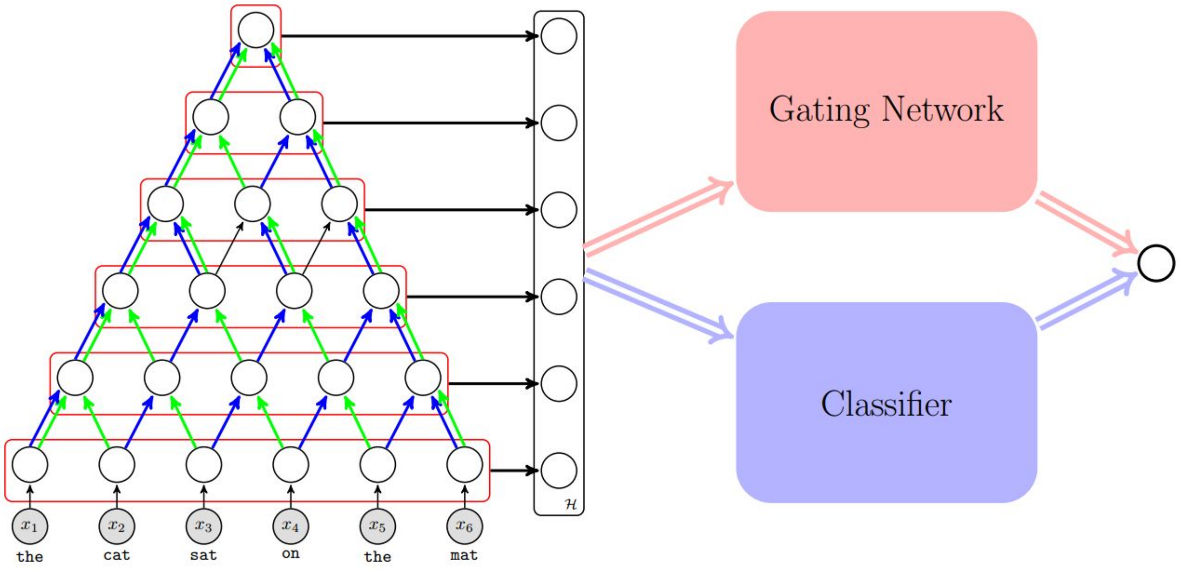
Hierachical ConvNet [Conneau et al. 2017]

At every layer, a representations are computed by a max-pooling operation over the feature maps. The final representation u = [u1, u2, u3, u4] concatenates representations at different levels of the input sentence.



Should we stack more layers?

AdaSent [Zhao et al. 2015]

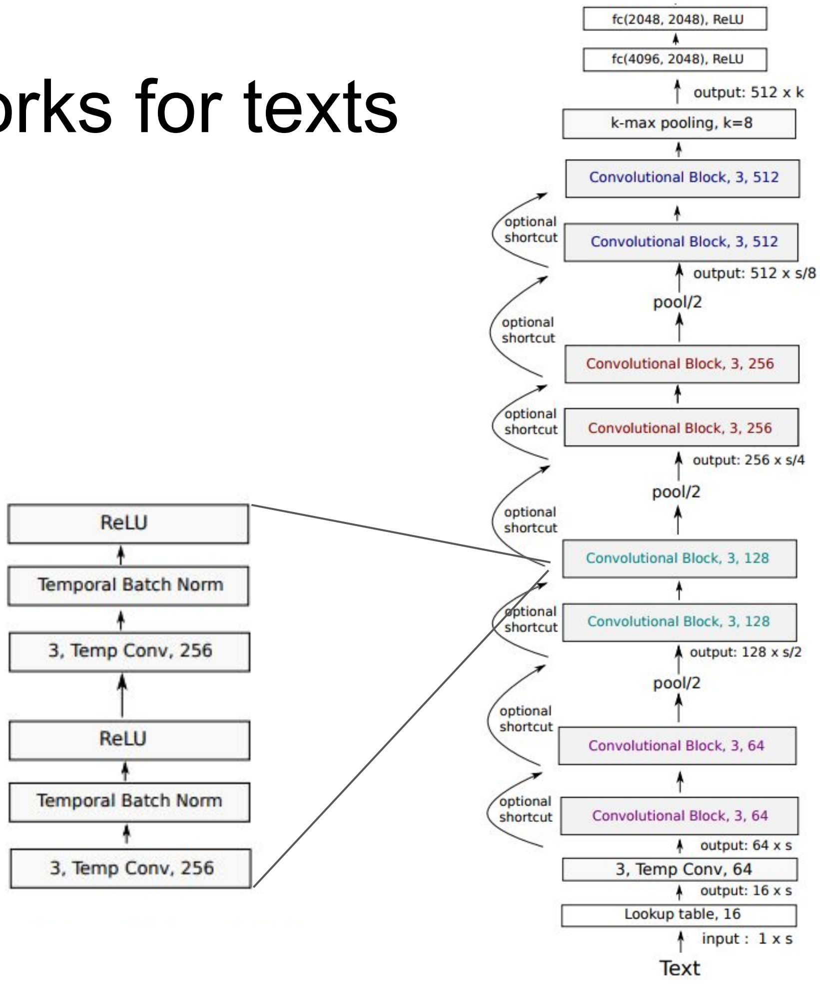


Deep convolutional networks for texts

Q: Can we get some quality points just stacking much more layers?

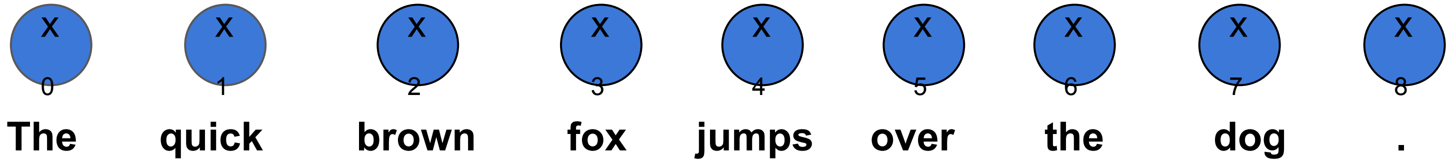
A: It does make sense in case character-level convolutional architectures.

VDCNN [Conneau et al. 2015] ~ ResNet-like network with 29 conv. layers



Recurrent NN for text classification

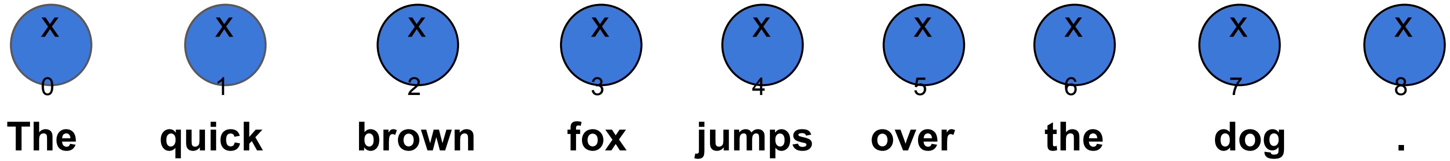
In a RNN Connections between nodes
form a directed graph along a sequence.



Recurrent NN for text classification

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

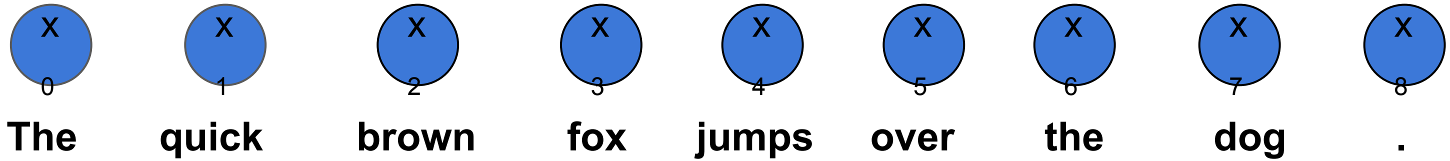
There are different types of recurrent units:
vanilla RNN, LSTM, GRU, MI-LSTM, peephole LSTM, …
But it's not important this time.



Recurrent NN for text classification

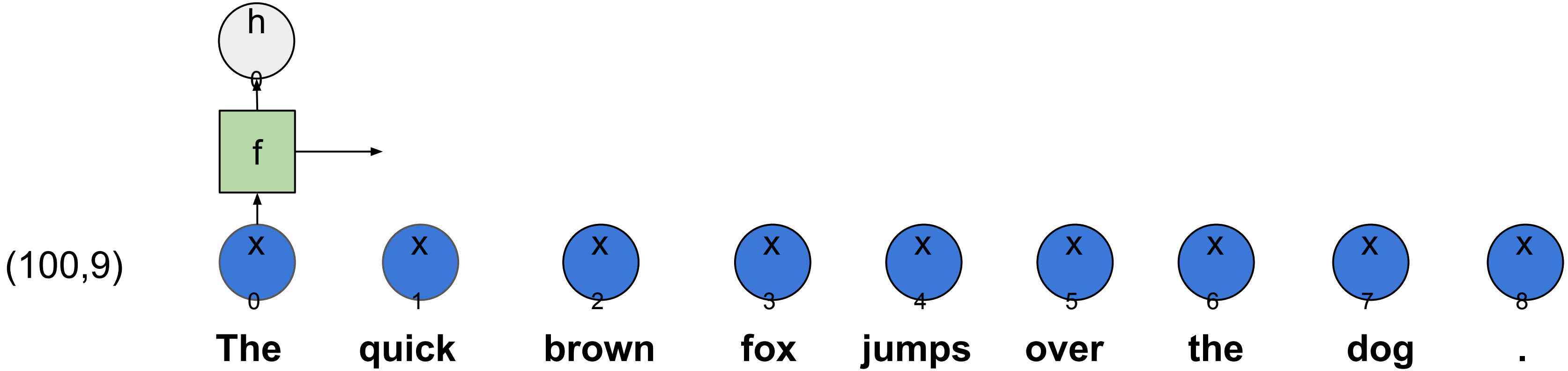
In a RNN Connections between nodes
form a directed graph along a sequence.

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



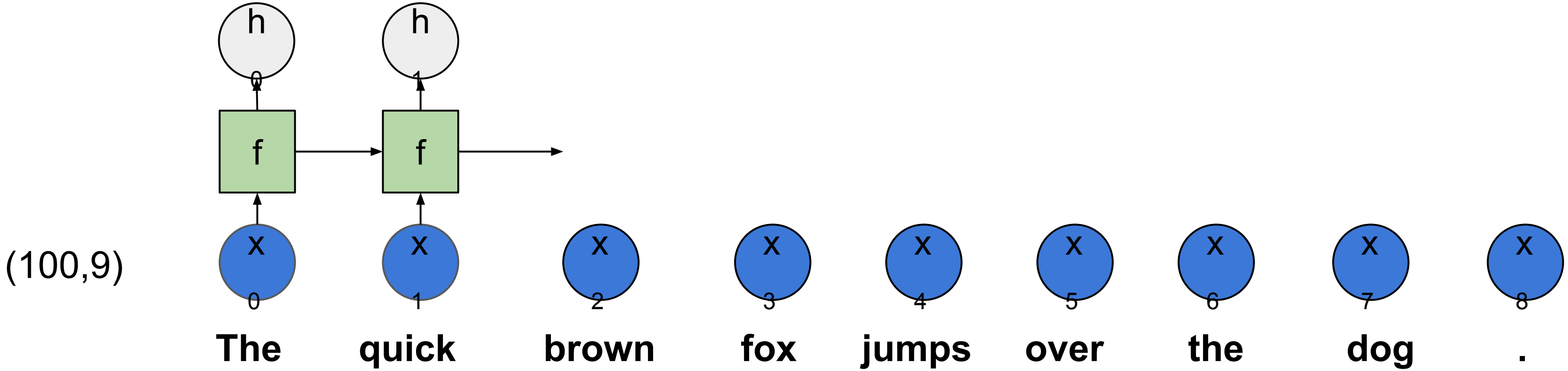
Recurrent NN for text classification

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



Recurrent NN for text classification

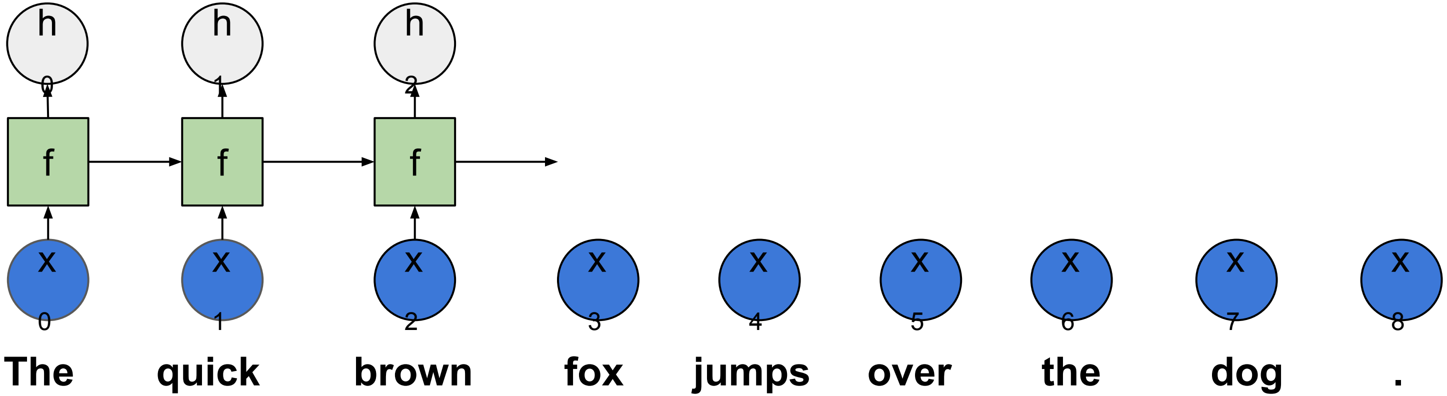
$$\mathbf{h}_t = f_w(\mathbf{h}_{t-1}, \mathbf{x}_t)$$



Recurrent NN for text classification

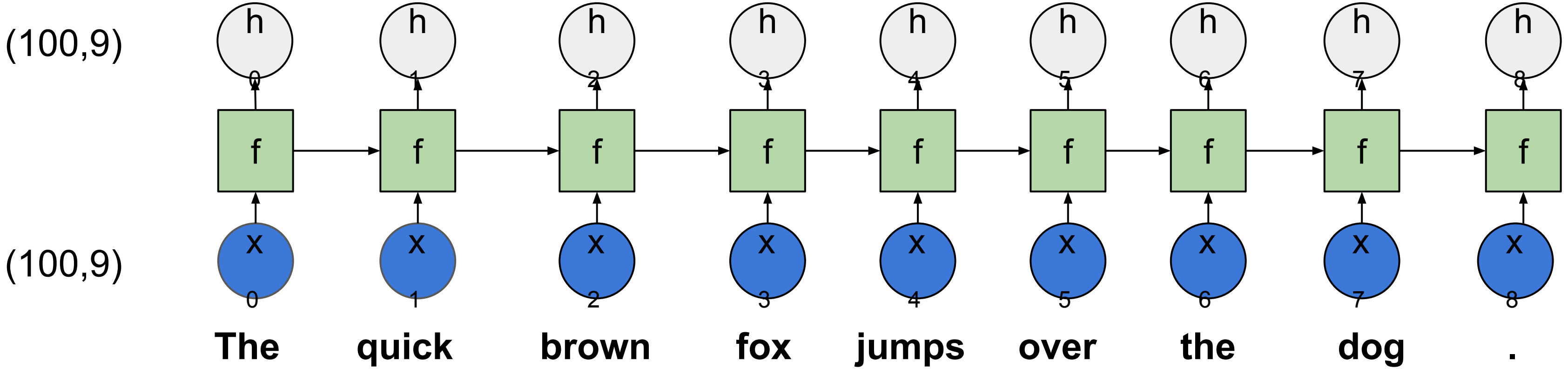
$$\mathbf{h}_t = f_w(\mathbf{h}_{t-1}, \mathbf{x}_t)$$

(100,9)



Recurrent NN for text classification

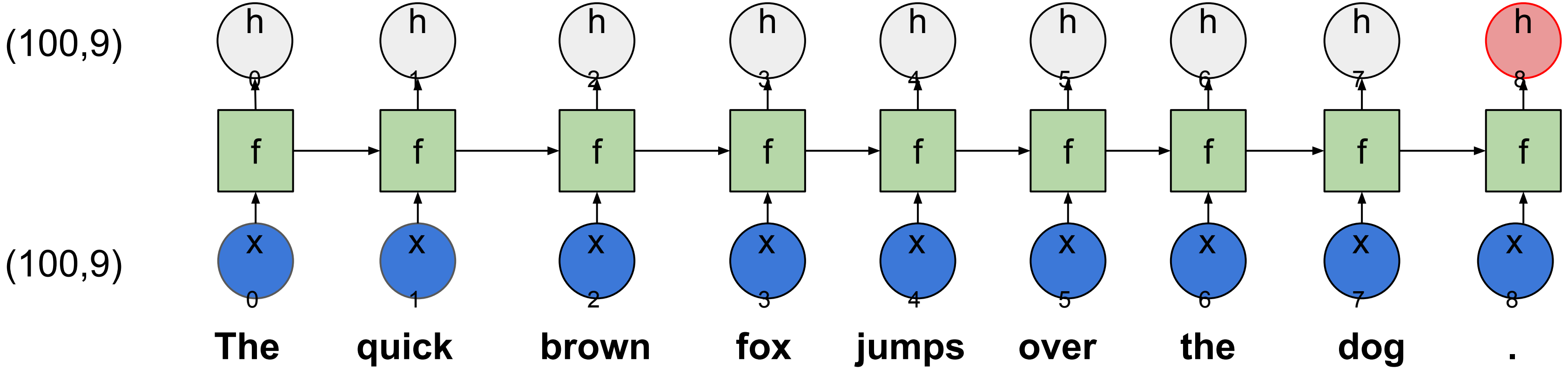
$$\mathbf{h}_t = f_w(\mathbf{h}_{t-1}, \mathbf{x}_t)$$



Recurrent NN for text classification

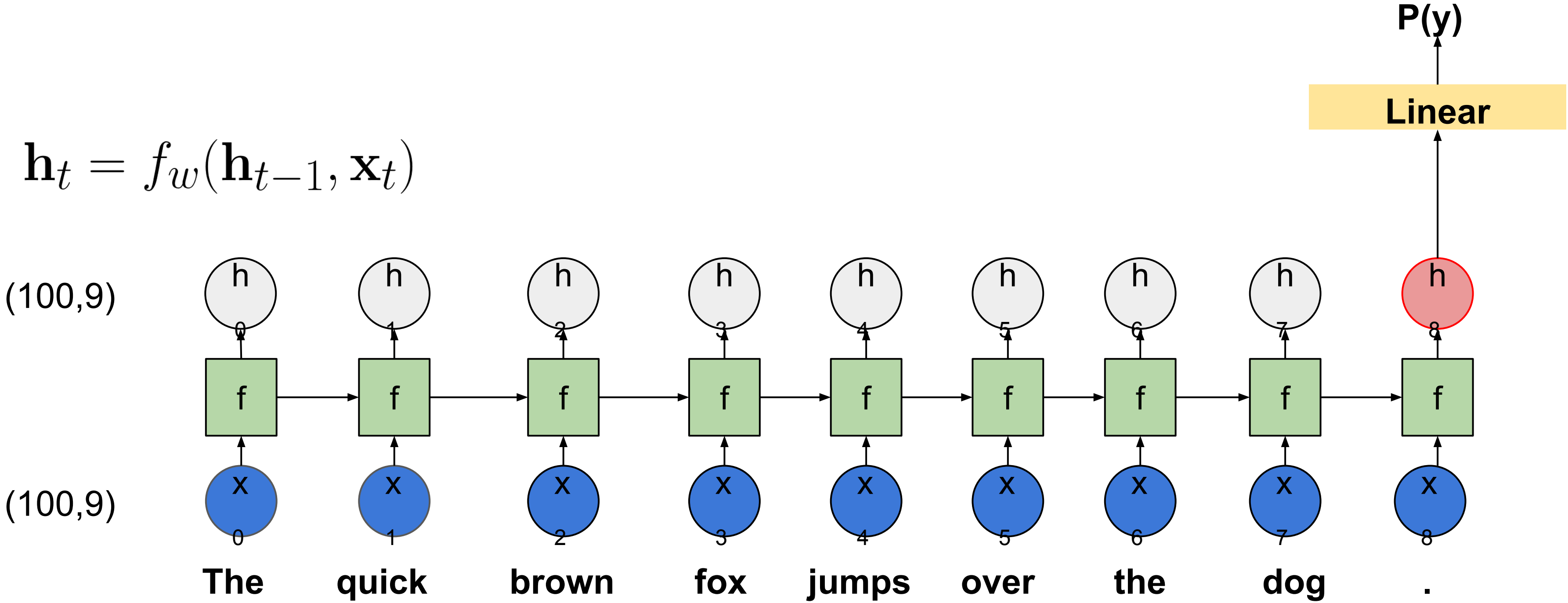
$$\mathbf{h}_8 = f(f(f(...(f(\mathbf{0}, \mathbf{x}_0)), \mathbf{x}_6), \mathbf{x}_7), \mathbf{x}_8)$$

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



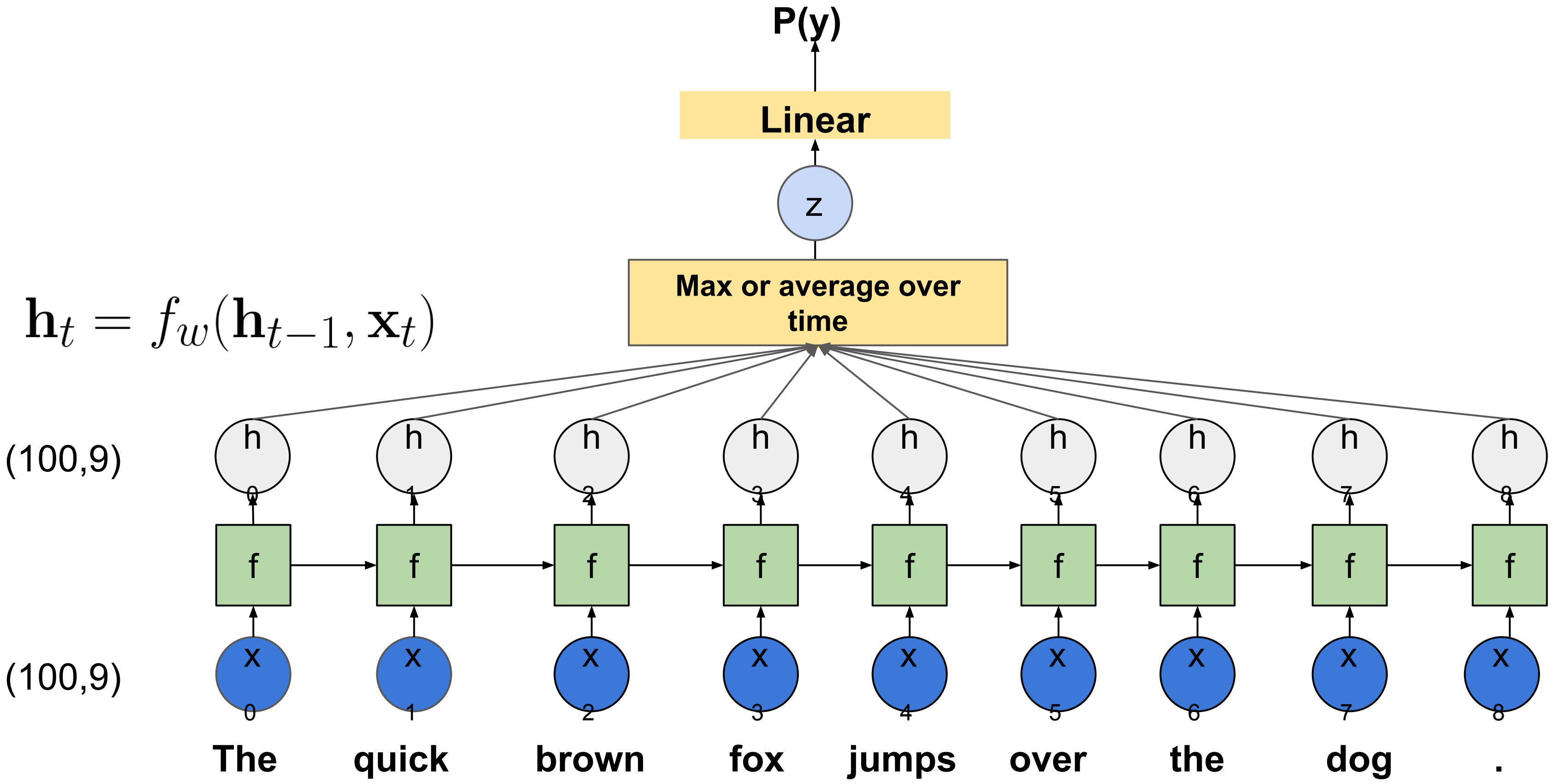
Recurrent NN for text classification

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

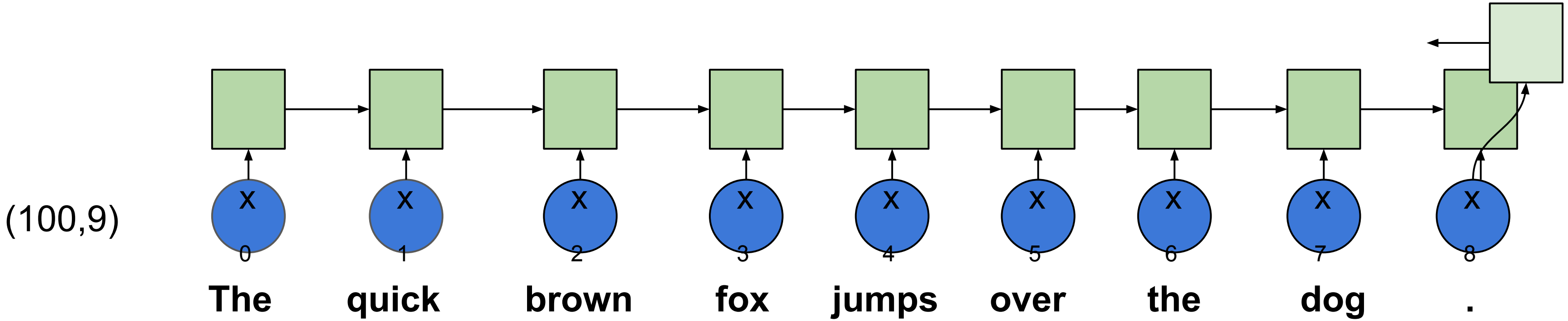


Recurrent NN for text classification

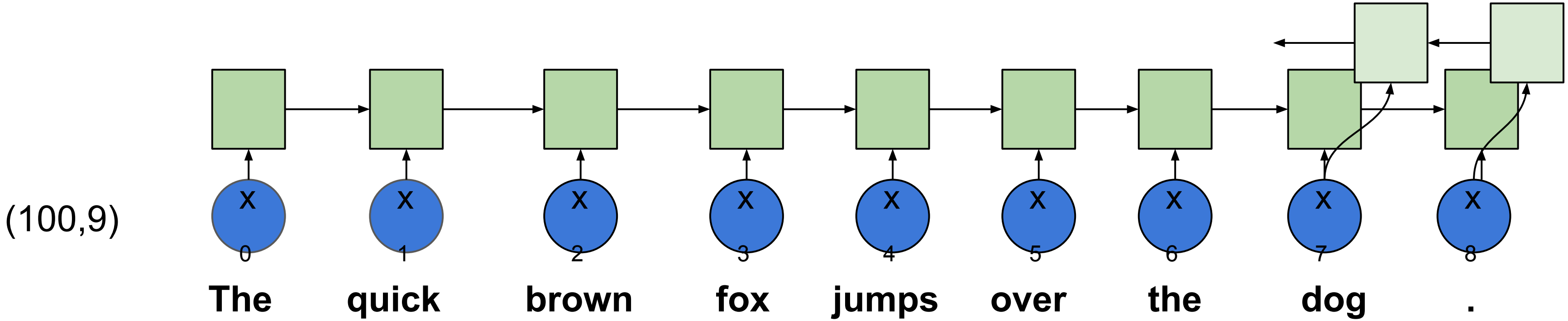
$$\mathbf{h}_t = f_w(\mathbf{h}_{t-1}, \mathbf{x}_t)$$



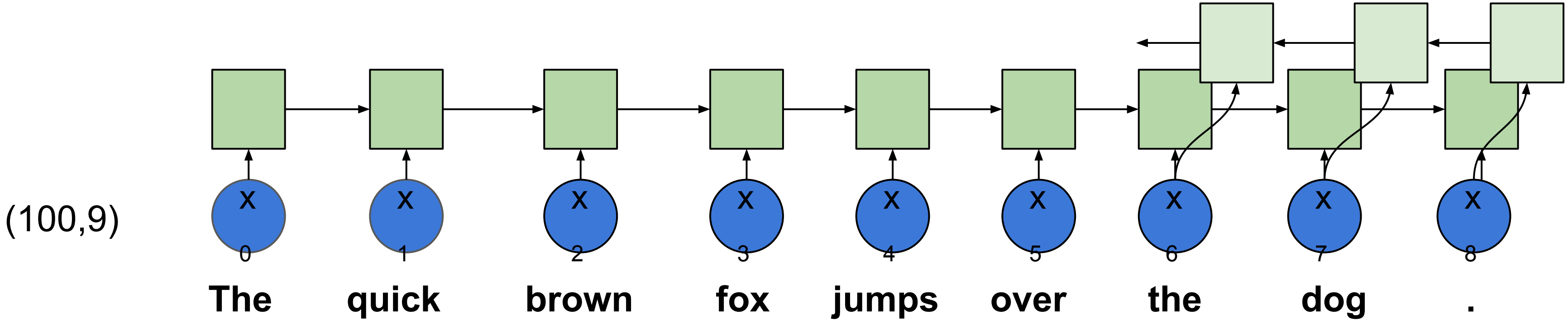
Recurrent NN for text classification



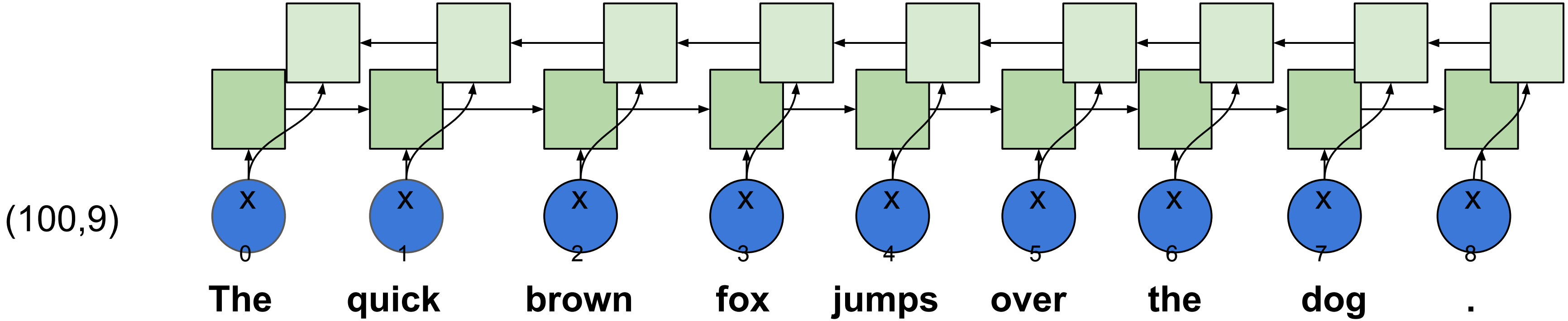
Recurrent NN for text classification



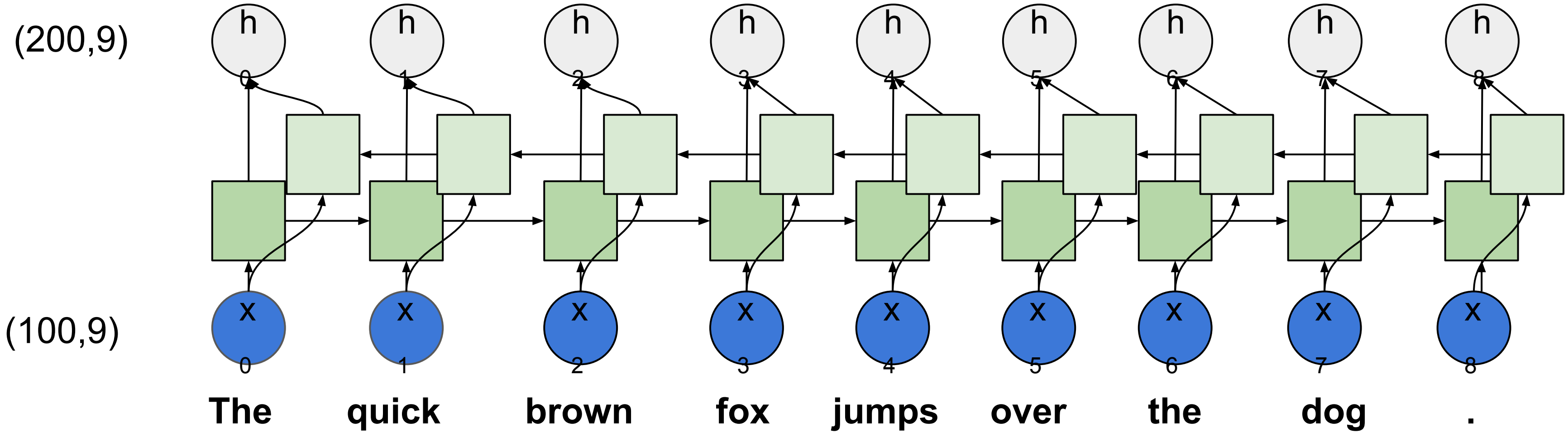
Recurrent NN for text classification



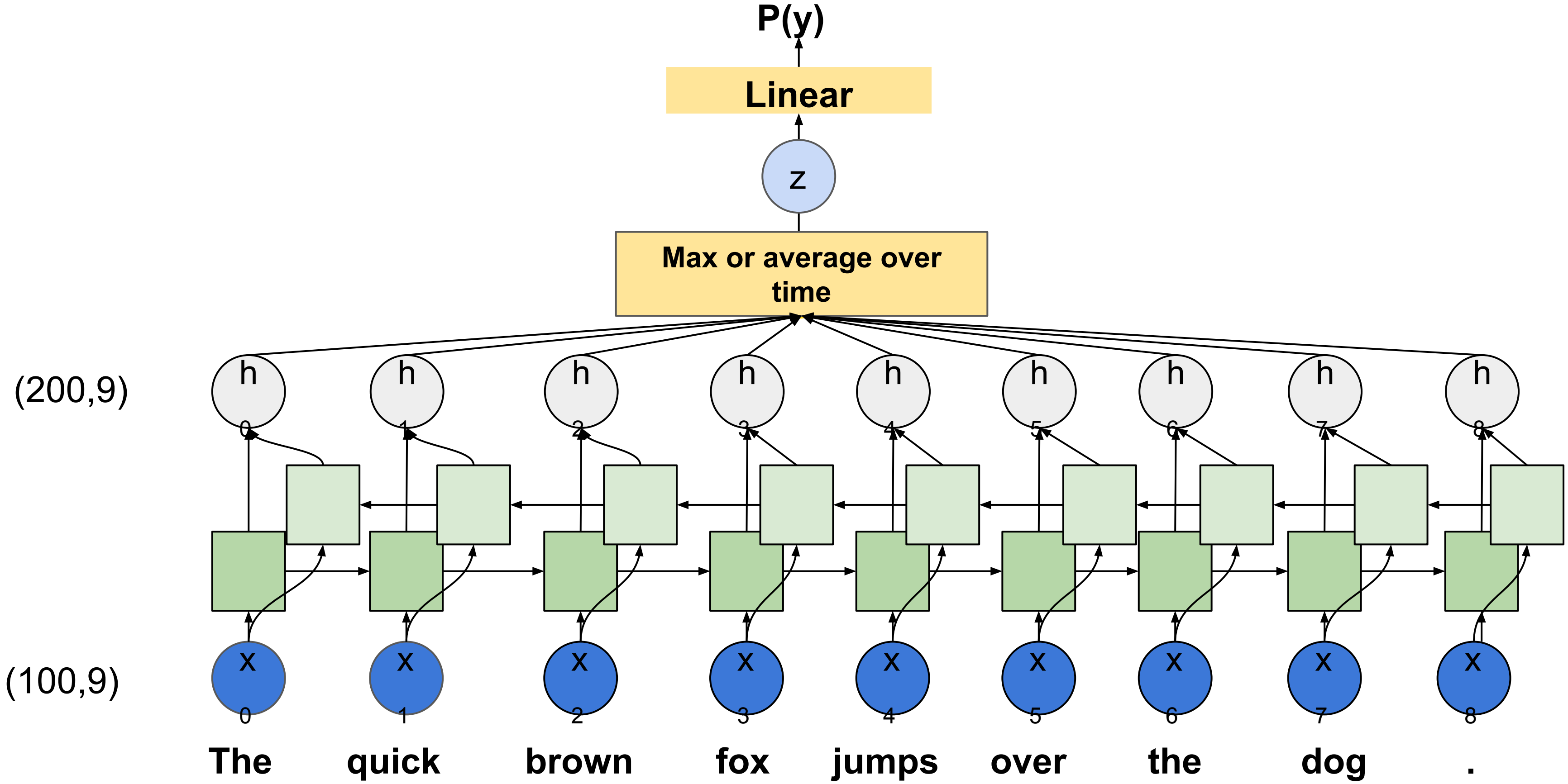
Recurrent NN for text classification



Recurrent NN for text classification



Recurrent NN for text classification



CNNs vs. RNNs

- With a lot of reservations RNNs demonstrates slightly better results on the benchmark classification tasks.
- CNNs work well on the tasks that can be reduced to keyword search. Keyword mean NEs, angry terms and so on.
- Also, RNNs have slower inference than CNNs. CNNs are easier to train.
- For RNN you need more data

[Yin et al 2017]

CNNs vs. RNNs

- With a lot of reservations RNNs demonstrates slightly better results on the benchmark classification tasks.
- CNNs work well on the tasks that can be reduced to keyword search. Keyword mean NEs, angry terms and so on.
- Also, RNNs have slower inference than CNNs. CNNs are easier to train.
- For RNN you need more data

It’s seems to be very task-dependent thing.
So you should try both options.

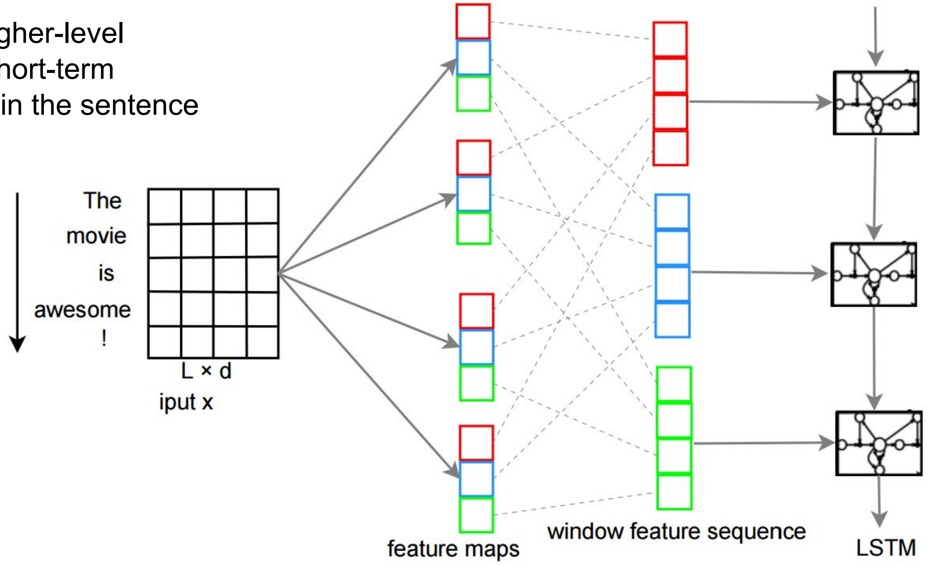
[Yin et al 2017]

We can combine them together though

C-LSTM [Zhou et al. 2015]

[conv.]->[LSTM]

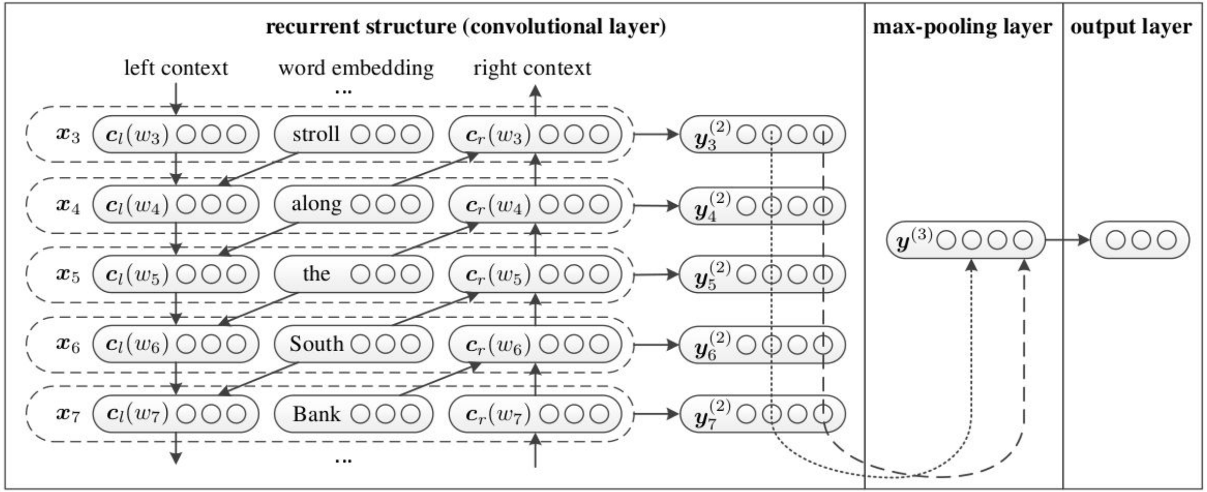
C-LSTM utilizes CNN to extract a sequence of higher-level phrase representations, and are fed into a long short-term memory recurrent neural network (LSTM) to obtain the sentence representation.



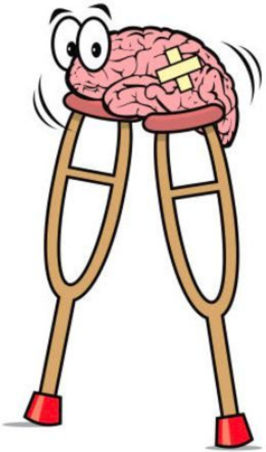
We can combine them together though

RCNN [Lai et al. 2015]

[Bi-RNN]->[conv.]



Hack of the day



Classical approaches vs. NNs

When should we use classical approaches?

Data augmentation in the image processing

An augmentation allows to increase train set and in theory it makes a network to be more robust to input variations.

Data augmentation in the image processing

An augmentation allows to increase train set and in theory it makes a network to be more robust to input variations.

1. Geometric transformations (flips, crops, rotations, non-linear transformations)

Data augmentation in the image processing

An augmentation allows to increase train set and in theory it makes a network to be more robust to input variations.

1. Geometric transformations (flips, crops, rotations, non-linear transformations)
2. Various input dropout schemes

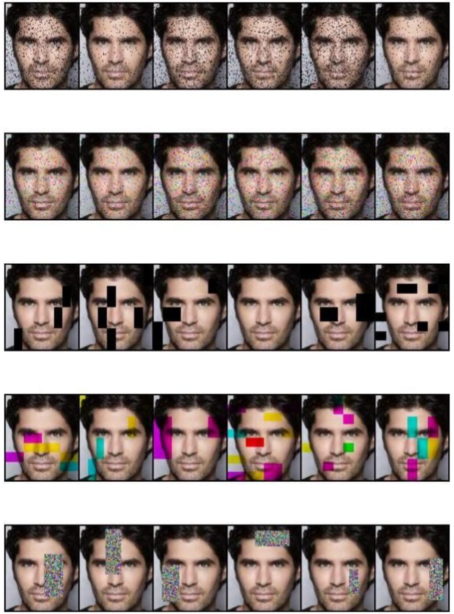
Dropout

Dropout per Channel

Coarse Dropout

Coarse Dropout per Channel

Random Erasing



What about NLP tasks?

Data augmentation in text classification

Texts can be augmented too.

1. Deformations (text pieces concatenation, paragraph reordering)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas suscipit et orci sed convallis. Ut augue libero, fringilla eget dui ut, rhoncus tincidunt lorem. Pellentesque ultricies fringilla venenatis. Maecenas nec sagittis neque.

+

Quisque bibendum ex risus, quis blandit metus posuere eu. Curabitur volutpat condimentum enim et vehicula. Vivamus urna nunc, commodo in lobortis vel, convallis at leo.

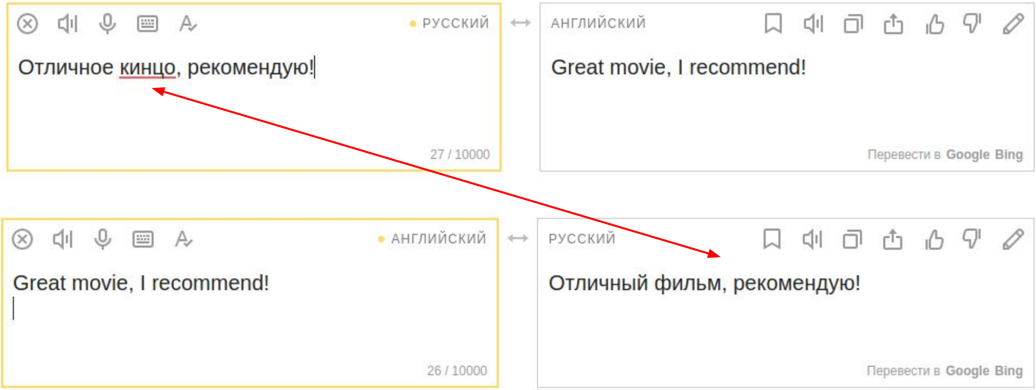
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas suscipit et orci sed convallis. Ut augue libero, fringilla eget dui ut, rhoncus tincidunt lorem. Pellentesque ultricies fringilla venenatis. Maecenas nec sagittis neque.

Quisque bibendum ex risus, quis blandit metus posuere eu. Curabitur volutpat condimentum enim et vehicula. Vivamus urna nunc, commodo in lobortis vel, convallis at leo.

Data augmentation in text classification

Texts can be augmented too.

1. Deformations (text pieces concatenation, paragraph reordering)
2. Reformulations (e.g. translation back and forth)



Data augmentation in text classification

Texts can be augmented too.

1. Deformations (text pieces concatenation, paragraph reordering)
2. Reformulations (e.g. translation back and forth)
3. Word dropout:
    a. A simple one, with <UNK>

*“Это был **худший** спектакль в моей жизни, просто ужас.”*

*“Это был **<UNK>** спектакль в моей жизни, просто ужас.”*

Data augmentation in text classification

Texts can be augmented too.

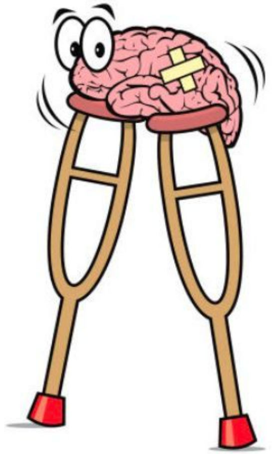
1. Deformations (text pieces concatenation, paragraph reordering)
2. Reformulations (e.g. translation back and forth)
3. Word dropout:
   a. A simple one, with <UNK>
   b. A smart one, using word semantic similarity information
   (e.g. using nearest neighbours in embedding space)

*“Это был **худший** спектакль в моей жизни, просто ужас.”*

*“Это был **никчемнеший** спектакль в моей жизни, просто ужас.”*

Hack of the day

Try to augment your data!



Thank you!