Negative sampling, Attention mechanism, 1D convolution and pre-trained feature extraction

Negative sampling for Word2Vec

Parameterization of the skipgram model

$$p(c|w;\theta) = \frac{e^{v_c \cdot v_w}}{\sum_{c' \in C} e^{v_{c'} \cdot v_w}}$$

EXPENSIVE COMPUTATION!!!!

We want to maximize this log-likelihood

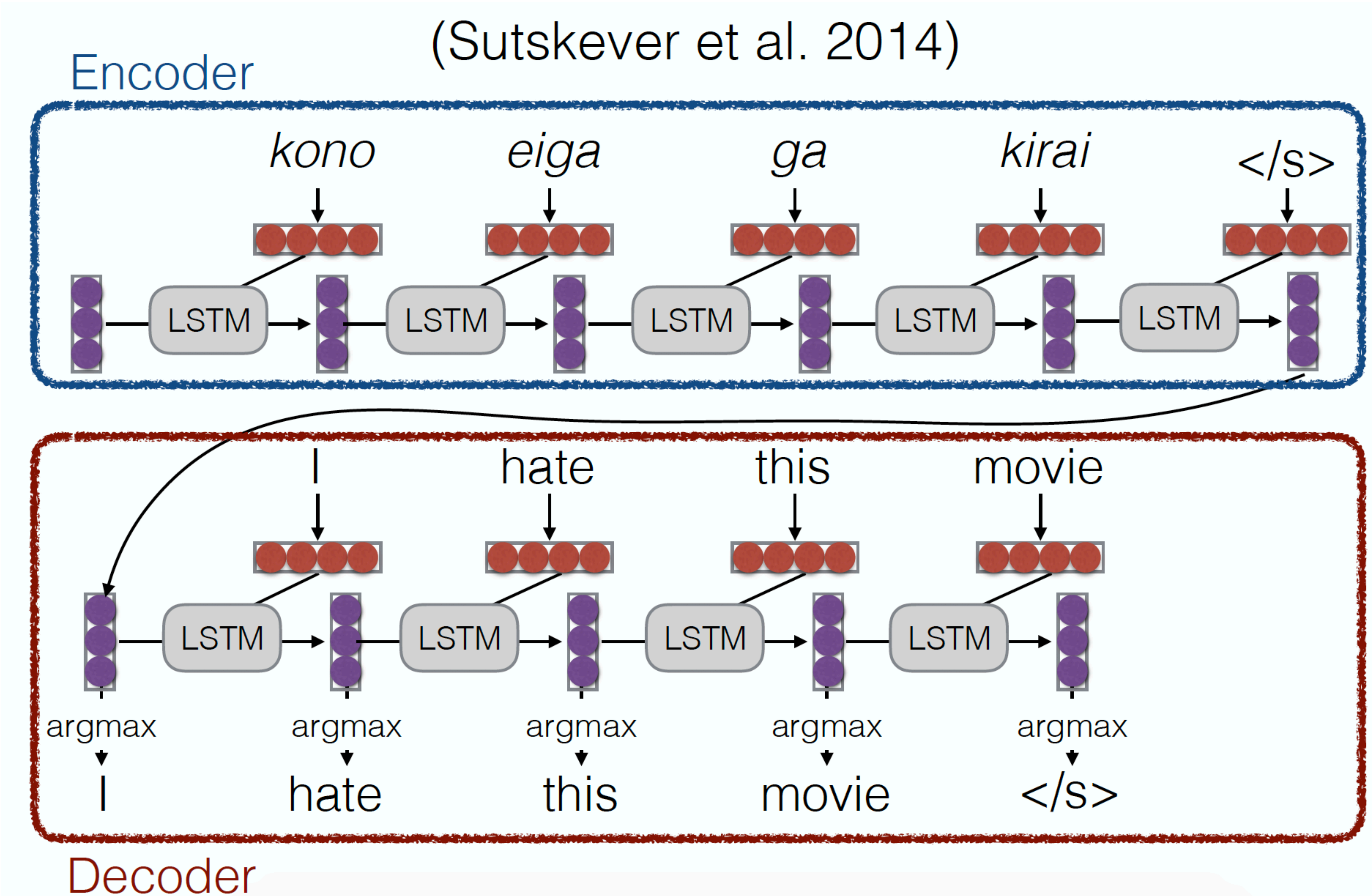
$$\arg\max_{\theta} \sum_{(w,c)\in D} \log p(c|w) = \sum_{(w,c)\in D} (\log e^{v_c \cdot v_w} - \log \sum_{c'} e^{v_{c'} \cdot v_w})$$

Negative sampling for Word2Vec

Instead of considering all the words in the vocabulary, consider a few "negative samples" for each "positive sample". Typically, we consider 5 negative samples. Randomly sample 5 negative (false) contexts for each positive (correct) (word, context) in the dataset. In the equation below, D' is the set of word and context which are invalid i.e., (w,c) ∈ D' means w never appears in the context c indicates the (w,c) ∈ D' is the set of all negative examples. The size of this set is much smaller than the original vocabulary size.

$$\arg\max_{\theta} \sum_{(w,c)\in D} \log \sigma(v_c \cdot v_w) + \sum_{(w,c)\in D'} \log \sigma(-v_c \cdot v_w)$$

Encoder-decoder Models



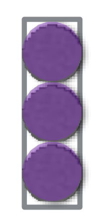
Sentence Representations

Problem!

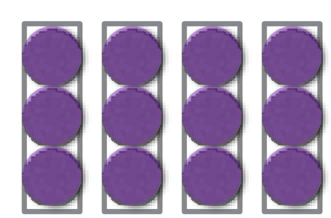
> "You can't cram the meaning of a whole %&!$ing sentence into a single $&!*ing vector!"
> — Ray Mooney

- But what if we could use multiple vectors, based on the length of the sentence.

this is an example →



this is an example →



Attention

Why attention?

- Look into distant features
- Combine all the features in the sequence to produce a better feature representation

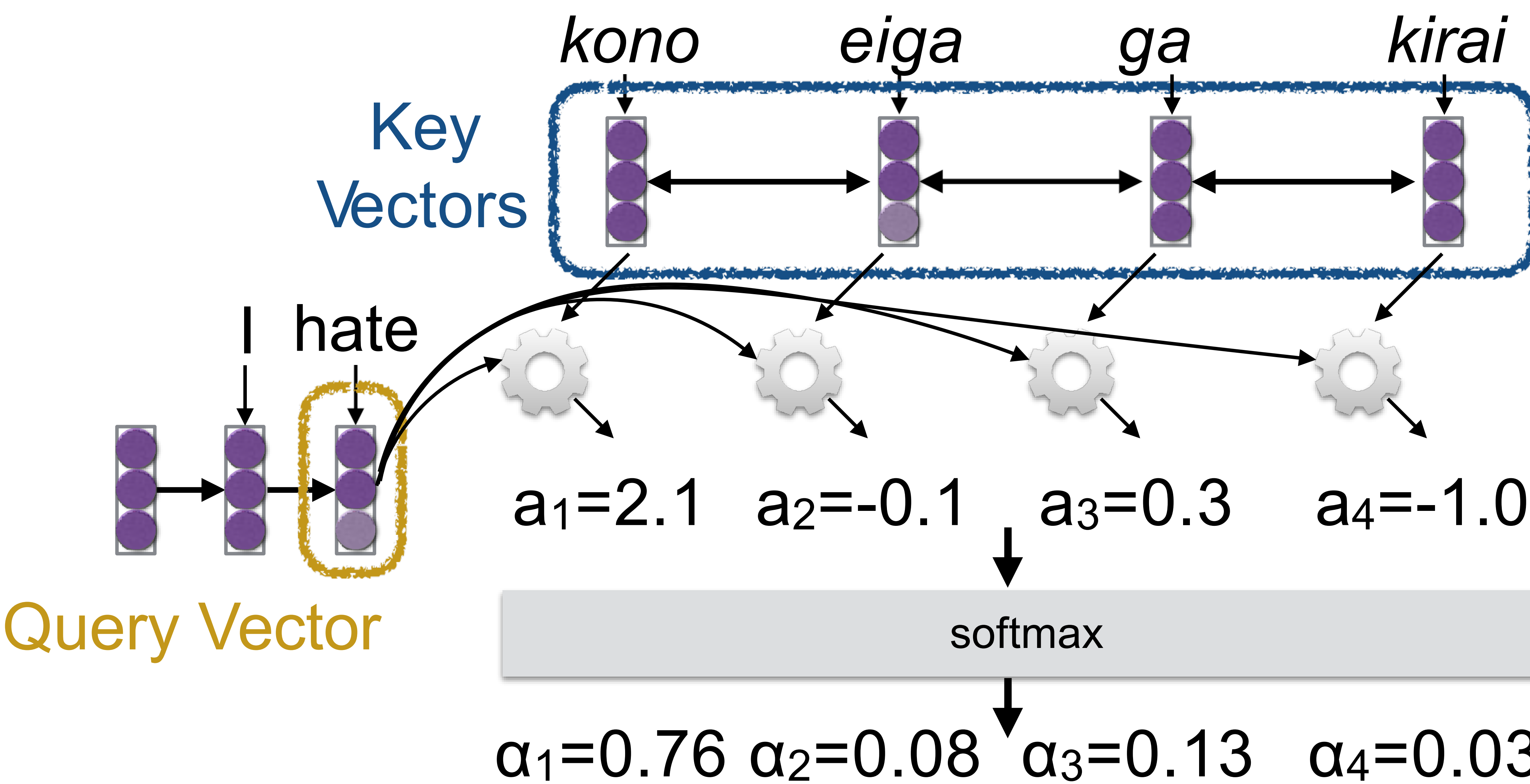
Basic Idea

(Bahdanau et al. 2015)

- Encode each word in the sentence into a vector
- When decoding, perform a linear combination of these vectors, weighted by “attention weights”
- Use this combination in picking the next word

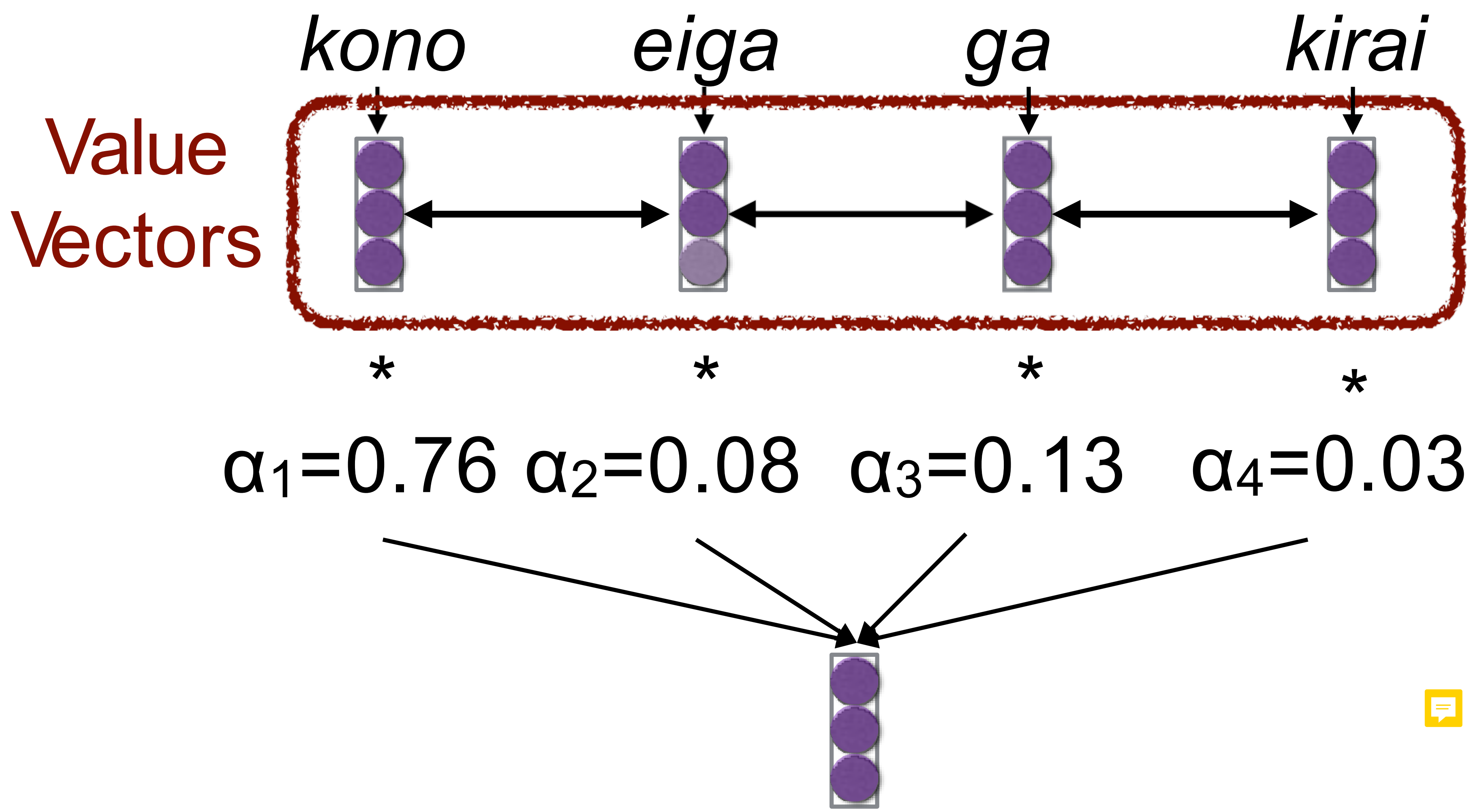
Calculating Attention (1)

- Use "query" vector (decoder state) and "key" vectors (all encoder states)
- For each query-key pair, calculate weight
- Normalize to add to one using softmax



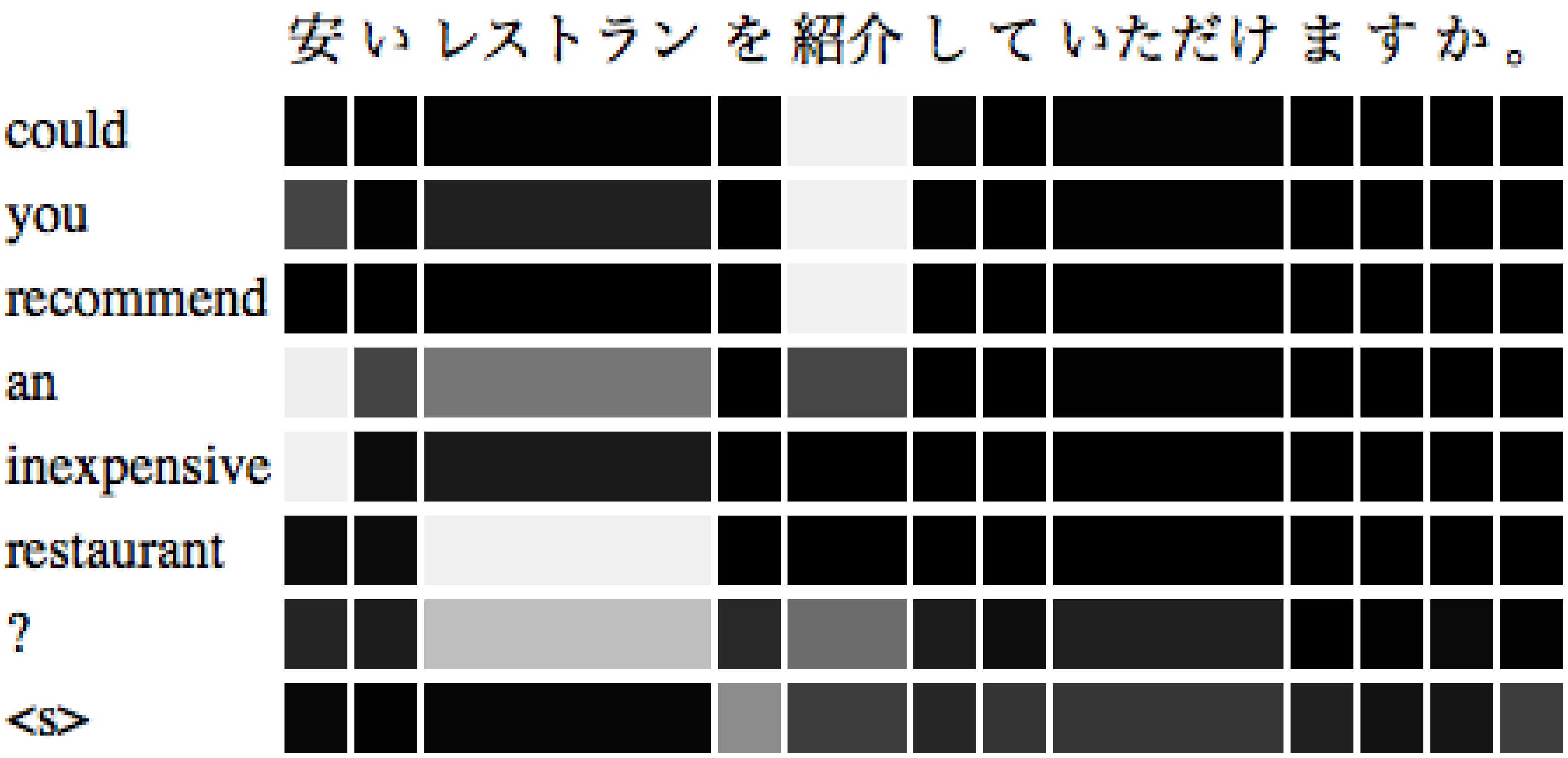
Calculating Attention (2)

- Combine together value vectors (usually encoder states, like key vectors) by taking the weighted sum



- Use this in any part of the model you like

A Graphical Example



Attention Score Functions (1)

- ***q*** is the query and ***k*** is the key

- **Multi-layer Perceptron** (Bahdanau et al. 2015)

$$a(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{w}_2^\intercal \tanh(W_1[\boldsymbol{q}; \boldsymbol{k}])$$

  - Flexible, often very good with large data

- **Bilinear** (Luong et al. 2015)

$$a(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{q}^\intercal W \boldsymbol{k}$$

Attention Score Functions (2)

- **Dot Product** (Luong et al. 2015)
  - No parameters! But requires sizes to be the same.

$$a(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{q}^{\intercal}\boldsymbol{k}$$

- **Scaled Dot Product** (Vaswani et al. 2017)
  - Problem: scale of dot product increases as dimensions get larger

$$a(\boldsymbol{q}, \boldsymbol{k}) = \frac{\boldsymbol{q}^{\intercal}\boldsymbol{k}}{\sqrt{|\boldsymbol{k}|}}$$

  - Fix: scale by size of the vector

Convolutional neural network for text classification

- The task: Given a textual training data, train a CNN for classification/regression.
- Do you find any similarity with the CNN applied on images?

Convolutional neural network for text classification

- Convert text to sequences

vocabulary - all unique words in a source of text
token - an integer value assigned to each word in the vocabulary

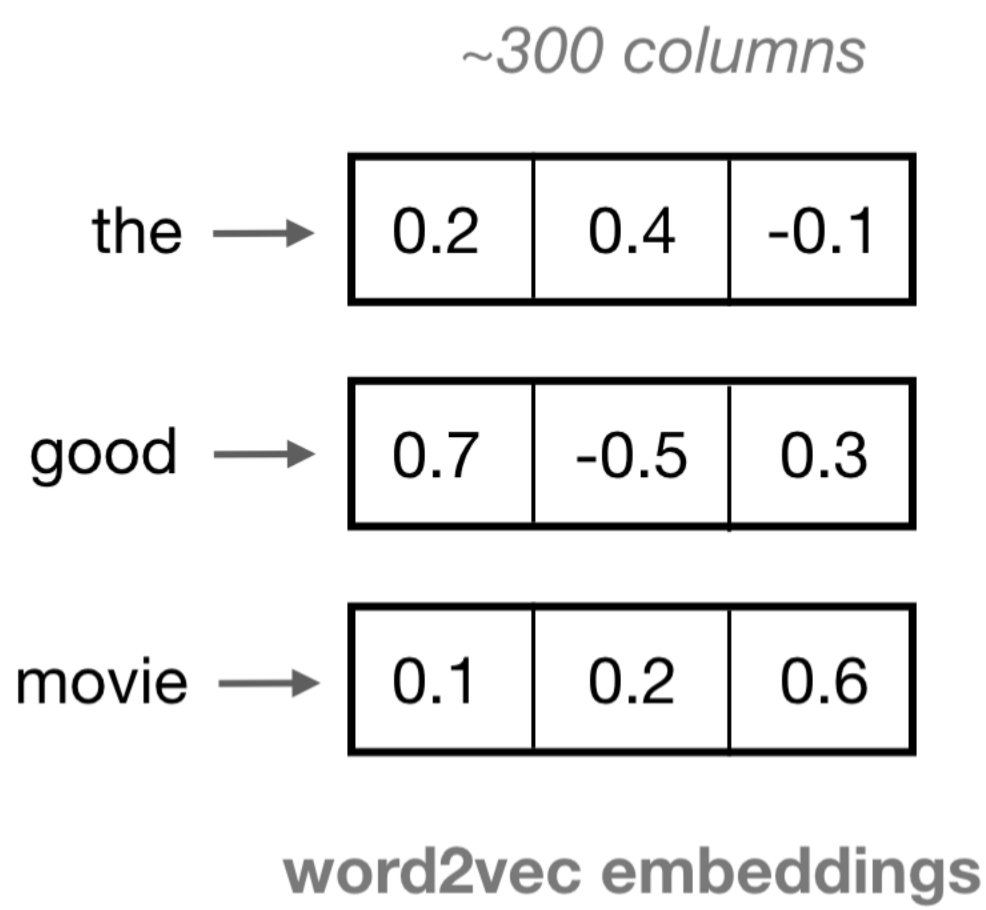
token dictionary

{'the': **0**, 'of': **1**, 'so': **2**, 'then': **3**, 'you': **4**, … 'learn': **3191**, … 'artificial': **30297**… }

sample text		**tokenized text**
"the pettiness of the whole situation"	→	`[0, 121241, 1, 0, 988, 25910]`

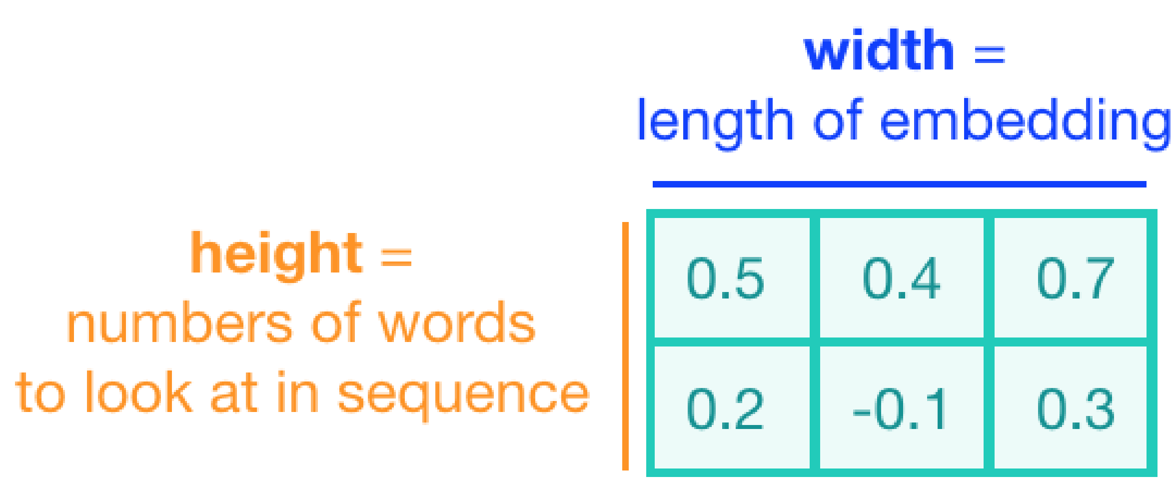
Convolutional neural network for text classification

- Use word embeddings



Convolutional neural network for text classification

- Convolutional kernels



Convolutional neural network for text classification

- Convolution over Word Sequences
  - Example – convolution over Bigrams.

the →	0.2	0.4	-0.1
good →	0.7	-0.5	0.3
movie →	0.1	0.2	0.6

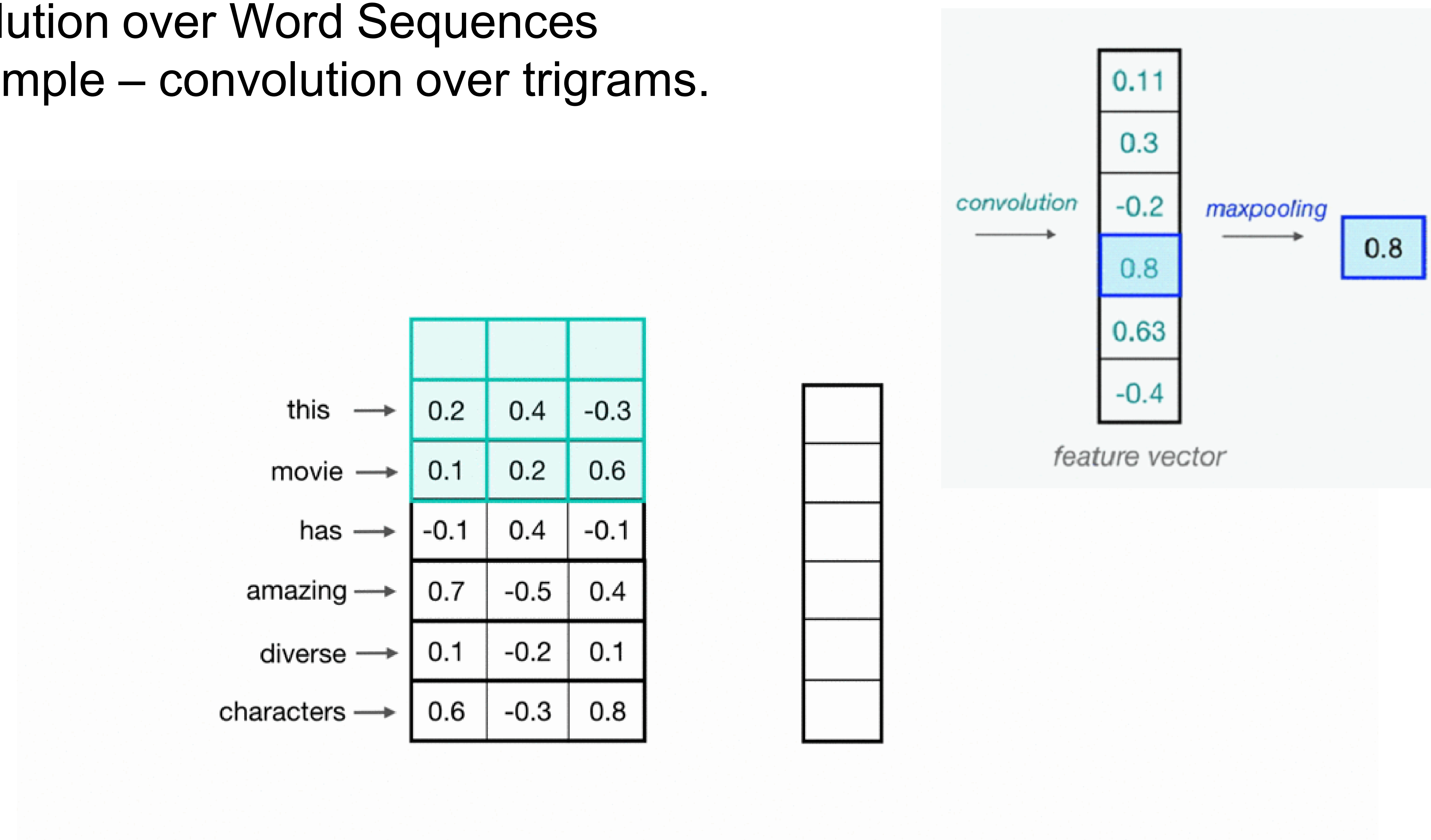
0.5	0.4	0.7
0.2	-0.1	0.3

convolutional kernel

$$0.5 * 0.7 + 0.4 * -0.5 + 0.7 * 0.3$$
$$+ 0.2 * 0.1 + -0.1 * 0.2 + 0.3 * 0.6$$
$$= \mathbf{0.54}$$

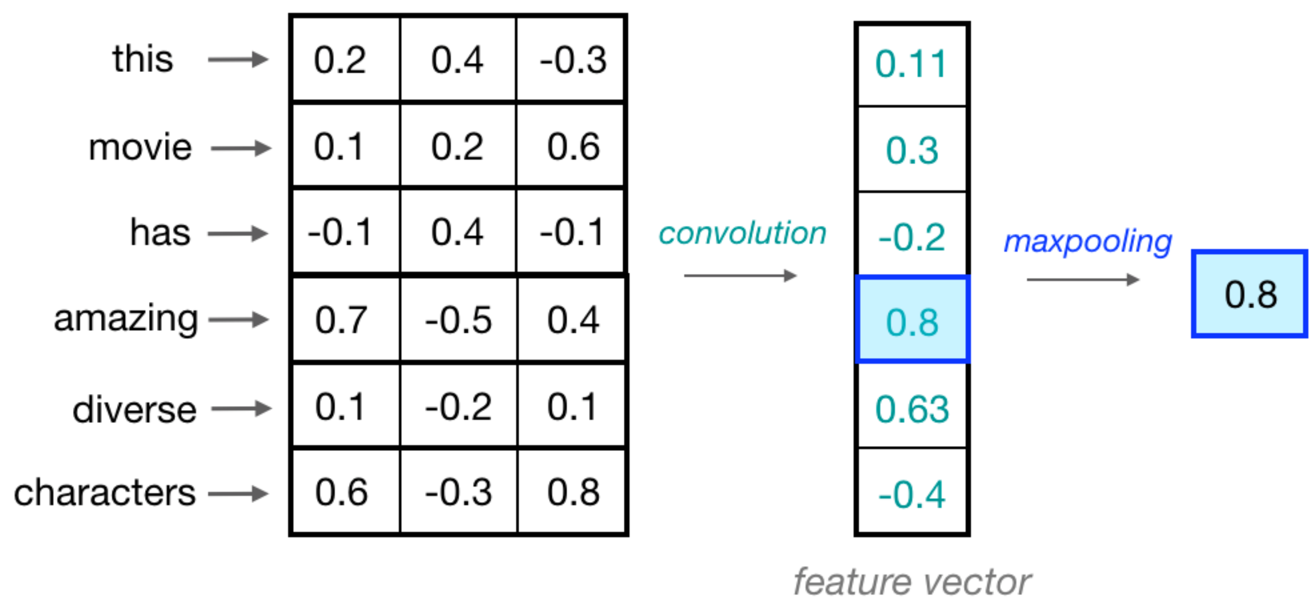
Convolutional neural network for text classification

- Convolution over Word Sequences
    - Example – convolution over trigrams.



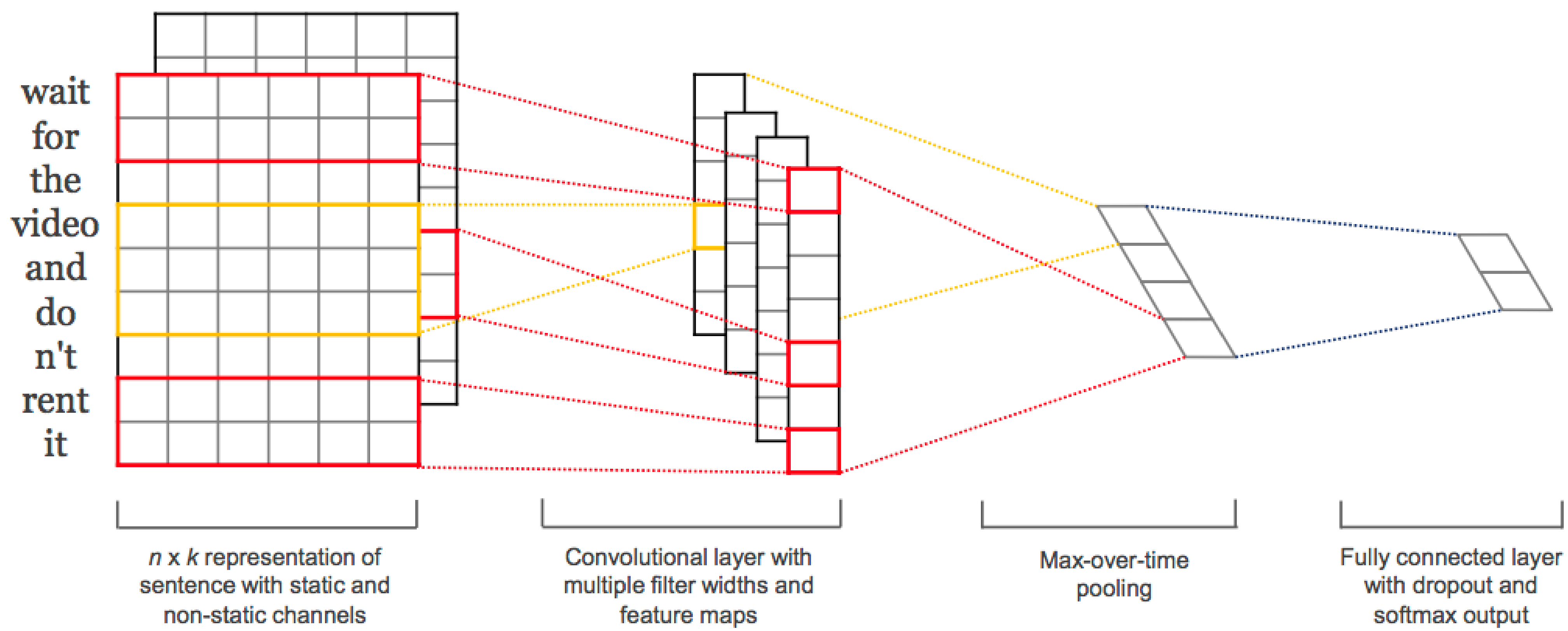
Convolutional neural network for text classification

- Maxpool

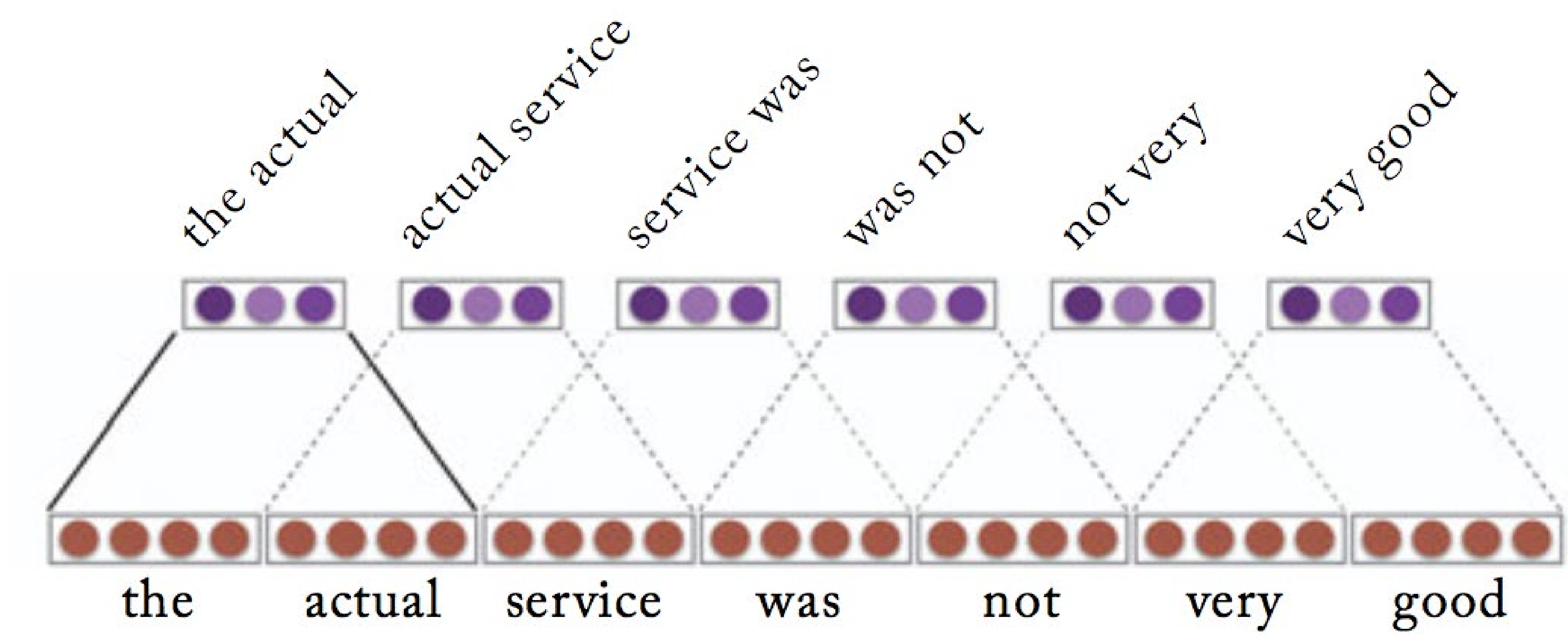


Convolutional neural network for text classification

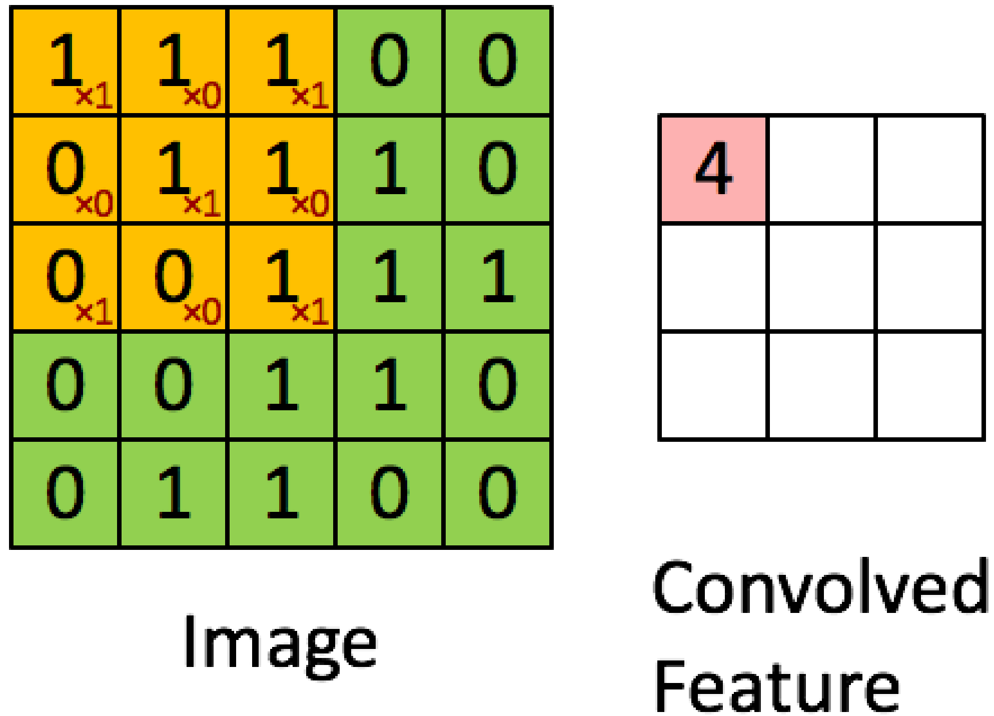
- The overall network



Convolutional neural network as N-gram feature extractor



CNN on images vs text



Pre-trained feature extraction

Suppose we want to solve a task A and we have a dataset D = {T_i, L_i} where T_i is the independent variable and L_i is the dependent variable i.e., label. We can train a supervised classifier on this dataset and use this network as a feature extractor for another task B. Note that task A and B are related but not the same task. How can we do it?

Pre-trained feature extraction

Consider the task of sarcasm extraction.