Variable Length Sequences N-gram features Convolutional Networks

Yoav Goldberg

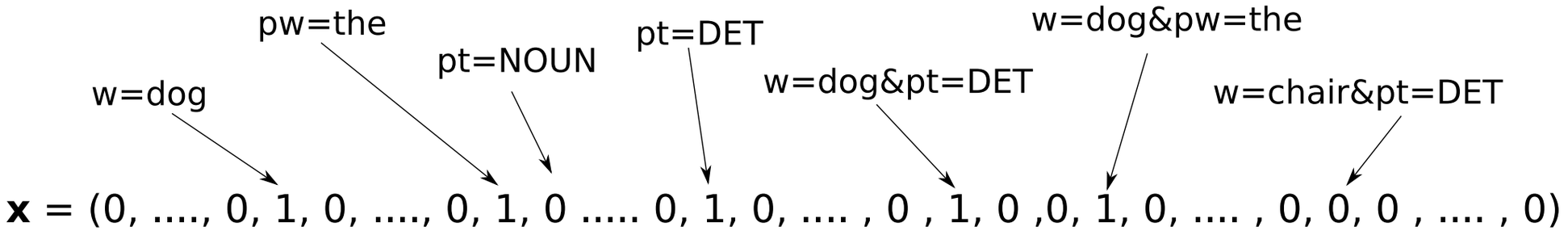
Reminder:
Word Embeddings

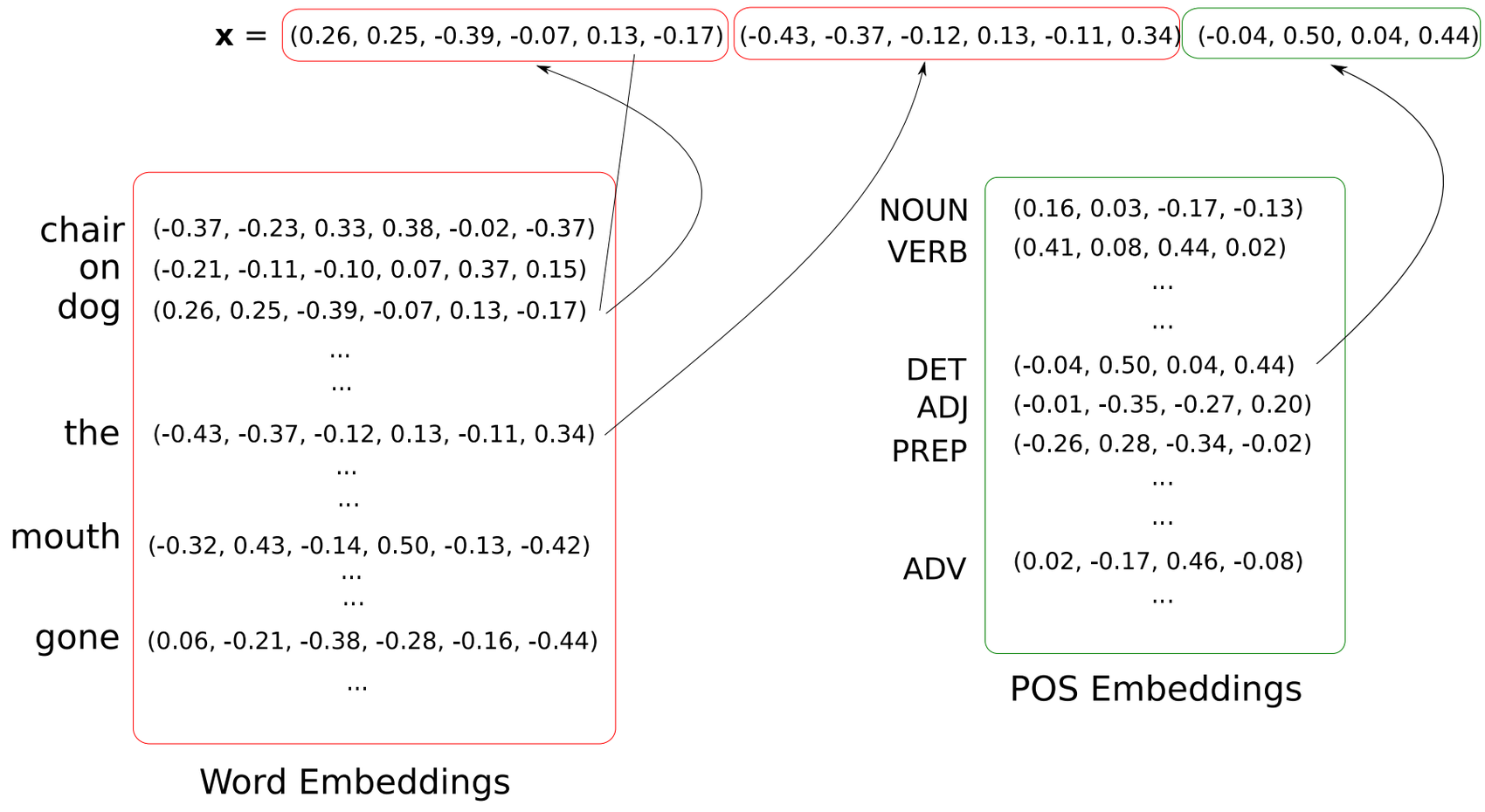
- Represent each vocabulary item as a d-dim vector.
- "Embedding matrix" stores all these vectors.
- Vectors are trained with the network.
- (relation to 1-hot X matrix).

"feature embeddings"

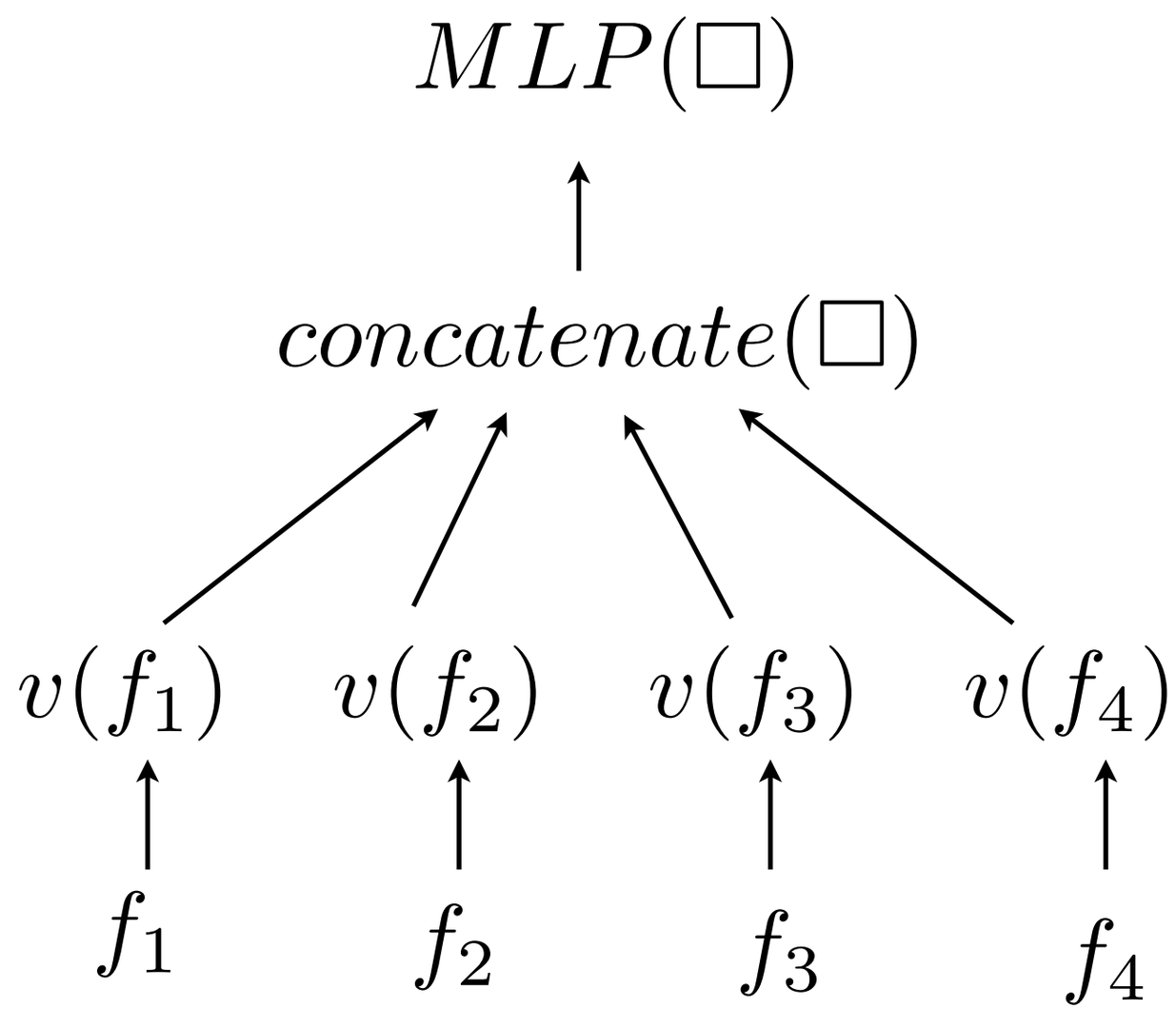
- Each feature is assigned a vector.
- The input is a combination of feature vectors.
- The feature vectors are **parameters of the model** and are trained jointly with the rest of the network.
- **Representation Learning**: similar features will receive similar vectors.

"feature embeddings"

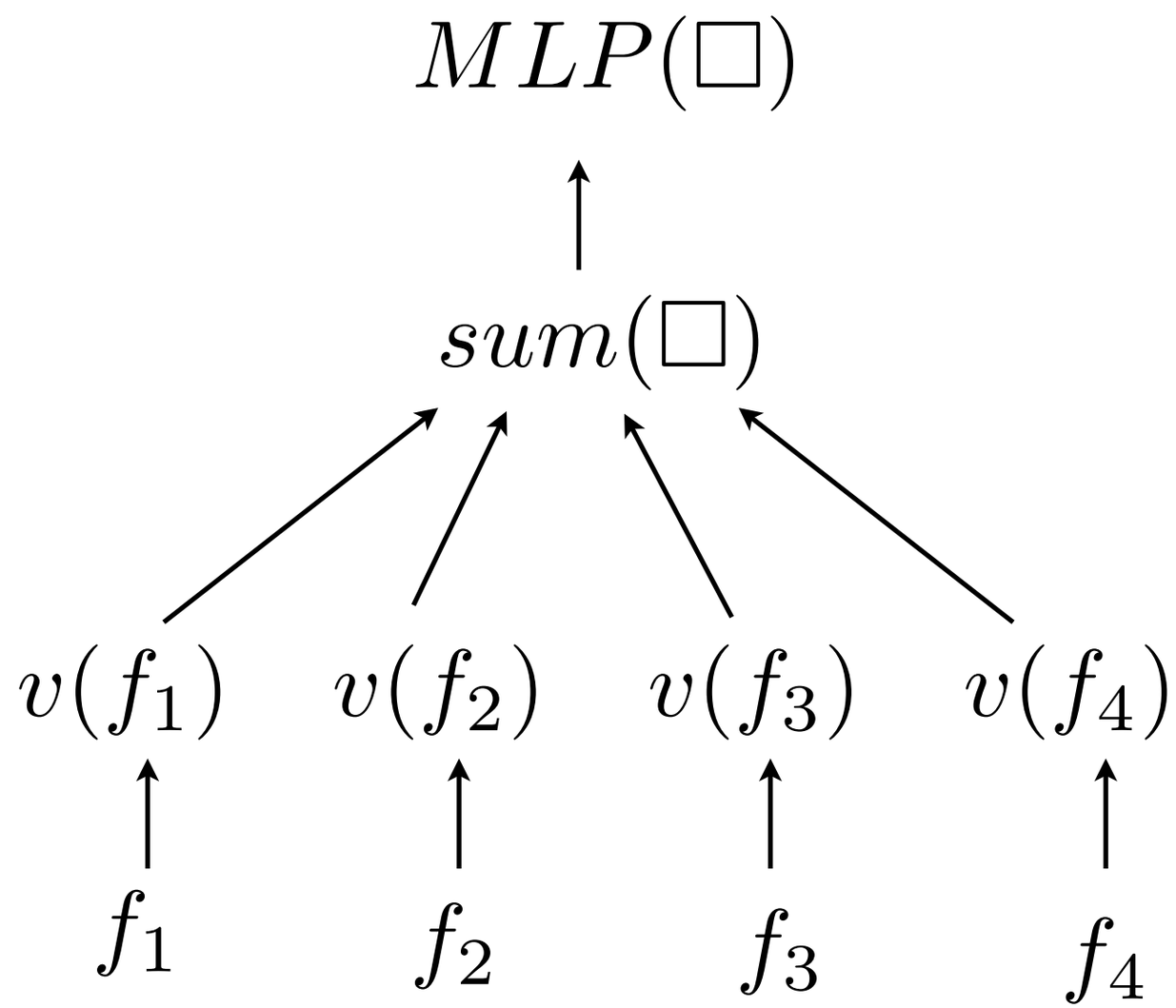




"feature embeddings"



"feature embeddings"



Concat vs. Sum

- **Concatenating** feature vectors: the "roles" of each vector is retained.

$$concat\left(v(”the”), v(”thirsty”), v(”dog”)\right)$$

prev word	current word	next word

- Different features can have vectors of different dim.

- Fixed number of features in each example (need to feed into a fixed dim layer).

Concat vs. Sum

- **Summing** feature vectors: "bag of features"

$$sum\left(v(”the”), v(”thirsty”), v(”dog”)\right)$$

word word word

- Different feature vectors should have same dim.
- Can encode arbitrary number of features.

Concat vs. Sum

- **Summing** feature vectors: "bag of features"

$$sum\left(v(”the”), v(”thirsty”), v(”dog”)\right)$$

word word word

- Different feature vectors should have same dim.
- Can encode **arbitrary number of features.**

Continuous Bag of Words (CBOW)

$$CBOW(f_1, ..., f_k) = \frac{1}{k} \sum_{i=1}^{k} v(f_i)$$

- a popular choice in document classification.
- can assign a different weight to each feature:

$$WCBOW(f_1, ..., f_k) = \frac{1}{\sum_{i=1}^{k} a_i} \sum_{i=1}^{k} a_i v(f_i)$$

Surprising Power of CBOW

$$CBOW(f_1, ..., f_k) = \frac{1}{k} \sum_{i=1}^{k} v(f_i)$$

Given CBOW vector and word vector,
can we predict if word is in cbow?

Surprising Power of CBOW

$$CBOW(f_1, ..., f_k) = \frac{1}{k} \sum_{i=1}^{k} v(f_i)$$

Given CBOW vector and two word vectors, can we predict which word appeared before the other?

Surprising Power of CBOW

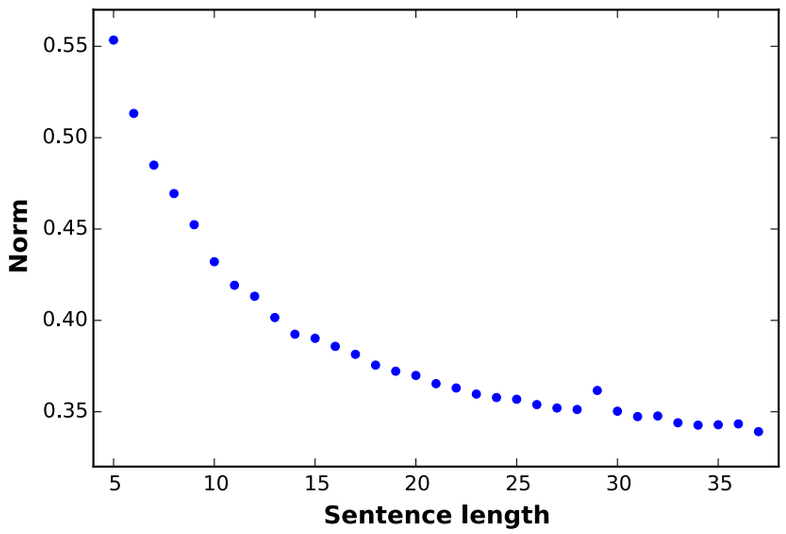
$$CBOW(f_1, ..., f_k) = \frac{1}{k} \sum_{i=1}^{k} v(f_i)$$

Given CBOW vector
can we predict sentence length?

Surprising Power of CBOW

$$CBOW(f_1, ..., f_k) = \frac{1}{k} \sum_{i=1}^{k} v(f_i)$$

Given CBOW vector
can we predict sentence length?



(b) Average embedding norm vs. sentence length for CBOW with an embedding size of 300.

how come?

Deep Unordered Composition Rivals Syntactic Methods for Text Classification

Mohit Iyyer,[1] **Varun Manjunatha,**[1] **Jordan Boyd-Graber,**[2] **Hal Daumé III**[1]
[1]University of Maryland, Department of Computer Science and UMIACS
[2]University of Colorado, Department of Computer Science
{miyyer,varunm,hal}@umiacs.umd.edu, Jordan.Boyd.Graber@colorado.edu

"Document Averaging Networks"

text classification

scores of labels

$softmax(\square)$

$CBOW(\square)$

$w_1, ..., w_n$

"neural bag of words"

scores of labels

$softmax(\square)$

$g^2(\mathbf{W}^2\square + \mathbf{b}^2)$

$g^1(\mathbf{W}^1\square + \mathbf{b}^1)$

$CBOW(\square)$

$w_1, ..., w_n$

"deep averaging network"

This is the log-linear model over bigrams from ass1! (see why?)

scores of labels

$\uparrow$

$softmax(\square)$

$\uparrow$

$CBOW(\square)$

$\uparrow$

$w_1, ..., w_n$

"neural bag of words"

scores of labels

$\uparrow$

$softmax(\square)$

$\uparrow$

$g^2(\mathbf{W}^2\square + \mathbf{b}^2)$

$\uparrow$

$g^1(\mathbf{W}^1\square + \mathbf{b}^1)$

$\uparrow$

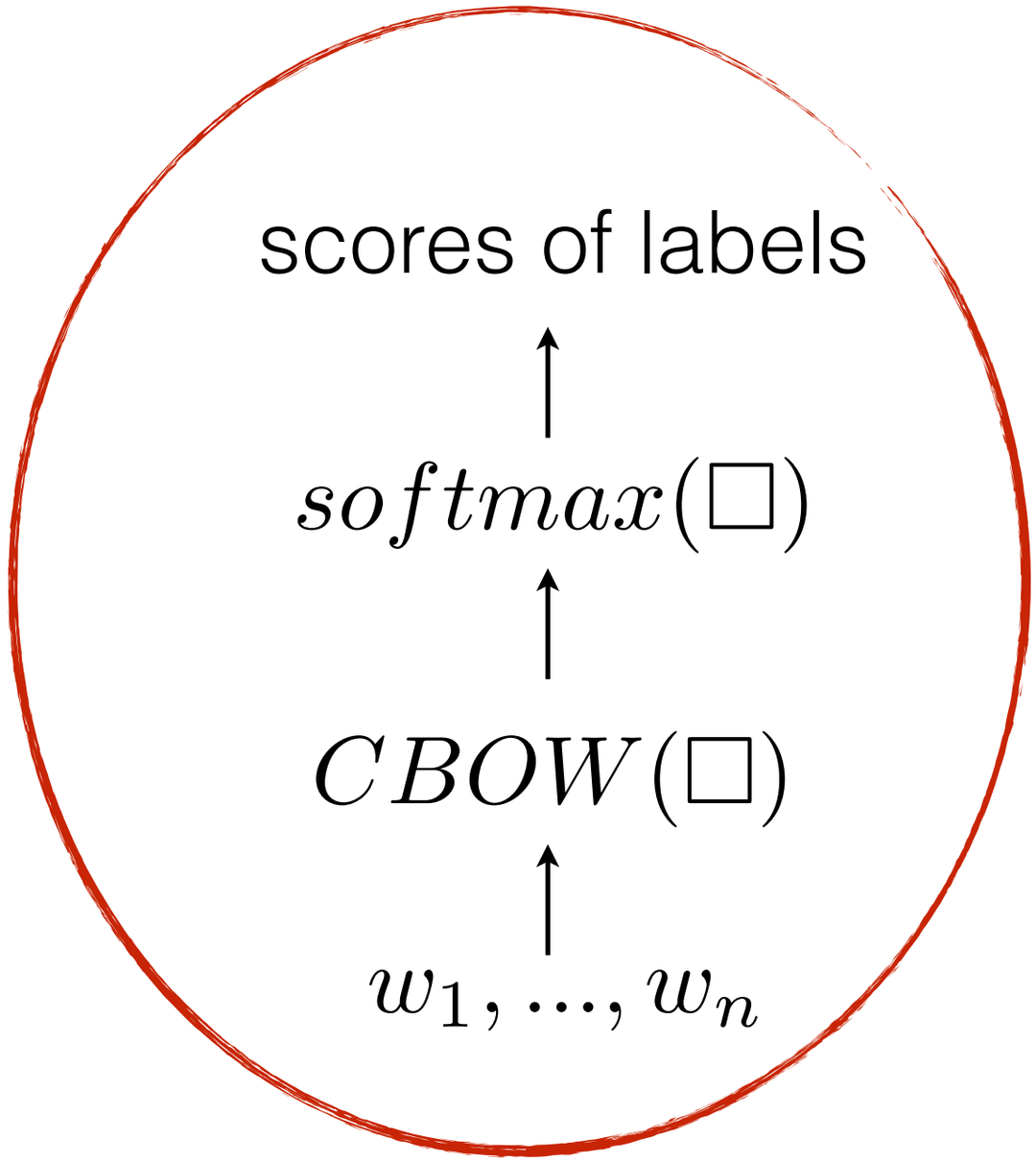
$CBOW(\square)$

$\uparrow$

$w_1, ..., w_n$

"deep averaging network"

Consider Assignment 1's model:



"neural bag of words"

If each feature is bigram, works great.

Moving to unigrams, large drop.

Unigrams + MLP --> better but not like bigrams.

Why?

Importance of Ngrams

- While we can ignore global order in many cases...
- ... local ordering is still often very important.
- Local sub-sequences encode useful structures.

Importance of Ngrams

- While we can ignore global order in many cases...
- ... local ordering is still often very important.
- Local sub-sequences encode useful structures.

(so why not just assign a vector to each ngram?)

Hybrids

- We have a language model, and we want to condition on the entire sentence history.

- We can have "window" (concatenation)
    - but then we are restricted to fixed length.
- We can have "cbow" (sum)
    - but then we loose word order information.

what can we do?

Hybrids

- For predicting word k:
  - concat words k-1, k-2, k-3, k-4 into v1
  - sum words 1,...,k-5 into v2
  - concat v1 and v2.
- We have local history with order, and the unordered context of the entire history also.

Hybrids

- For predicting word k:
  - concat words k-1, k-2, k-3, k-4 into v1
  - sum words 1,...,k-5 into v2
  - concat v1 and v2.
- We have local history with order, and the unordered context of the entire history also.

another option: when summing words, give weight 1/2 to k-1, 1/3 to k-2, 1/4 to k-3, etc.

ConvNets

special architecture for local predictors

ConvNets

- CBOW allows encoding arbitrary length sequences, but loses all order information.

- Some local order (i.e. bigrams, trigrams) is informative. Yet, we do not care about exact position in the sequence. (think "good" vs. "not good")

- ConvNets (in language) allow to identify informative local predictors.

- Works by moving a shared function (feature extractor) over a sliding window, then pooling results.

ConvNets

- ConvNets have huge success in computer vision.
- It allows invariance to object position.
- It allows composing large predictors from small.

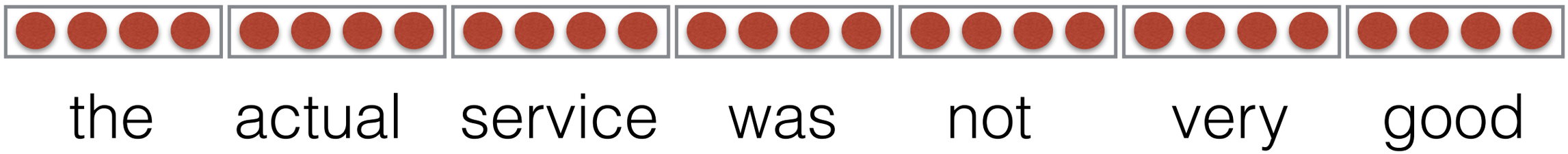
(you've seen them in 2d in ML class)

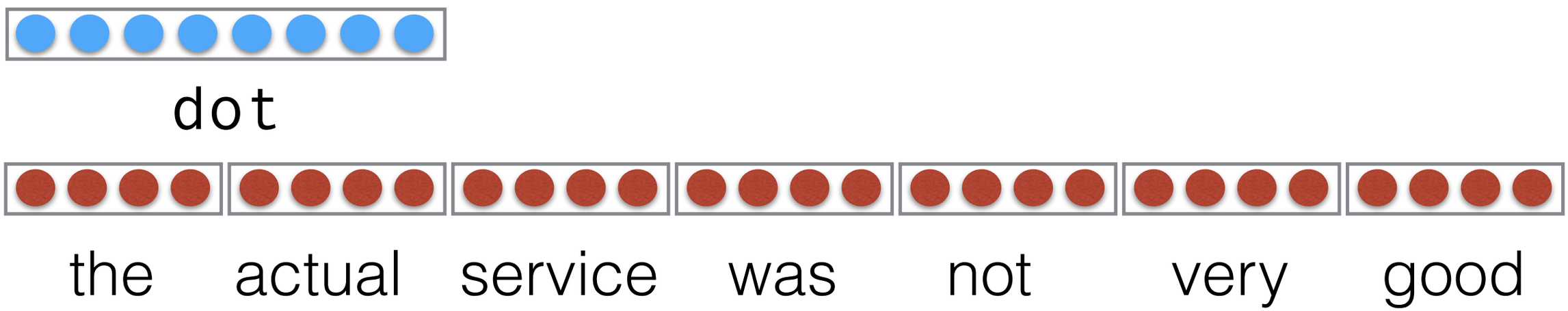
from

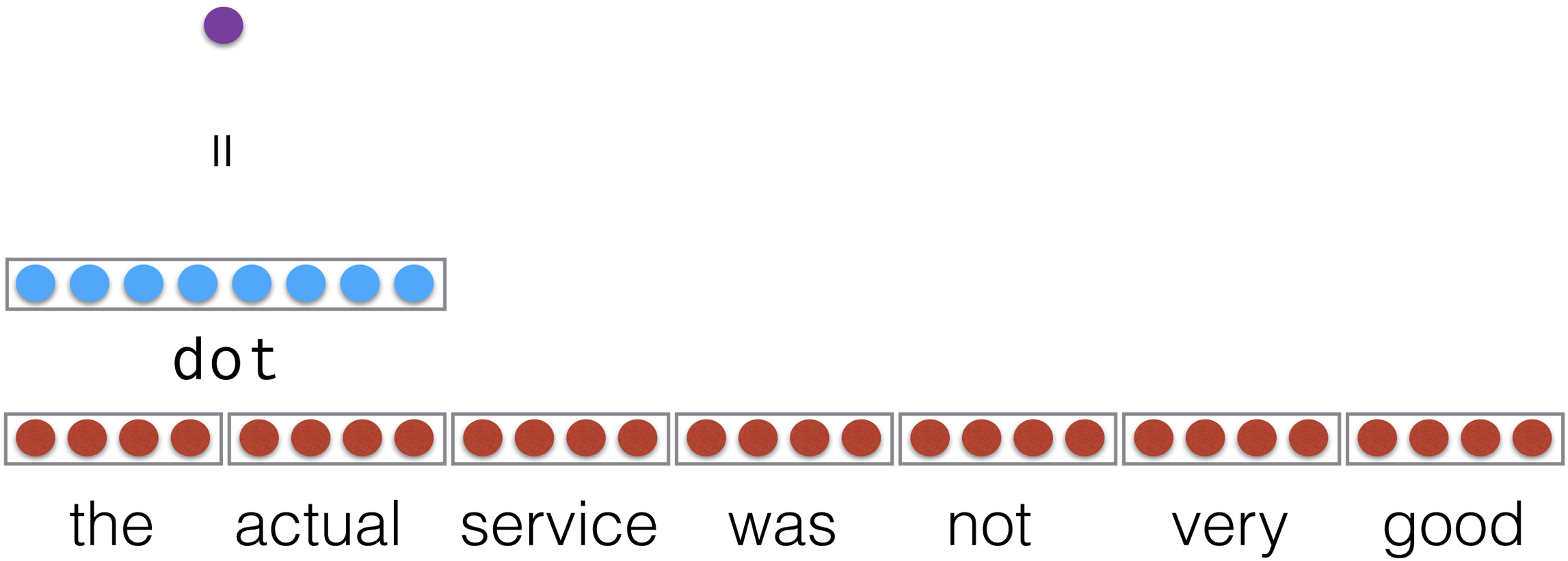
2d ConvNets

to

1d ConvNets


the
actual
service
was
not
very
good


dot
the
actual
service
was
not
very
good

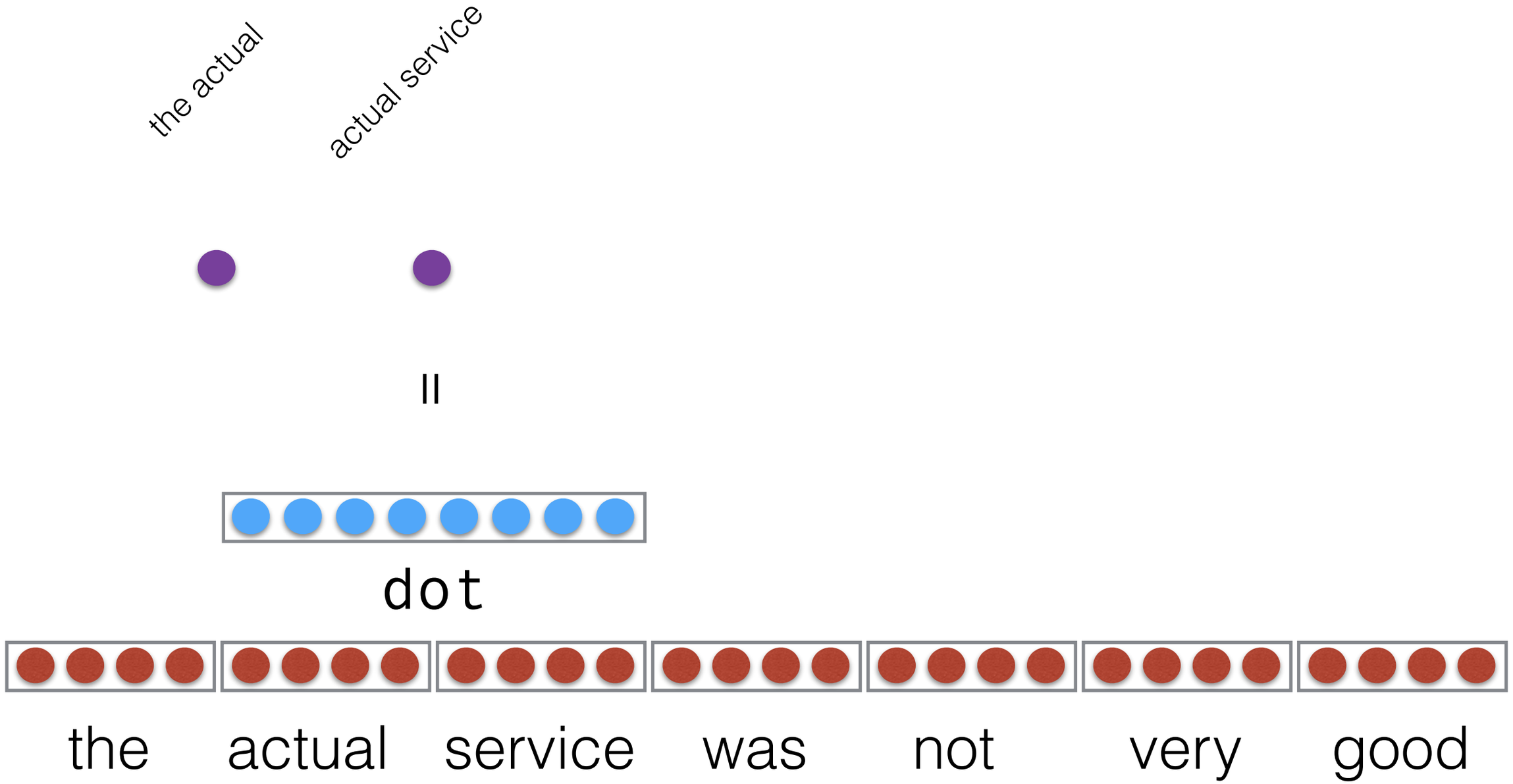

dot
the
actual
service
was
not
very
good

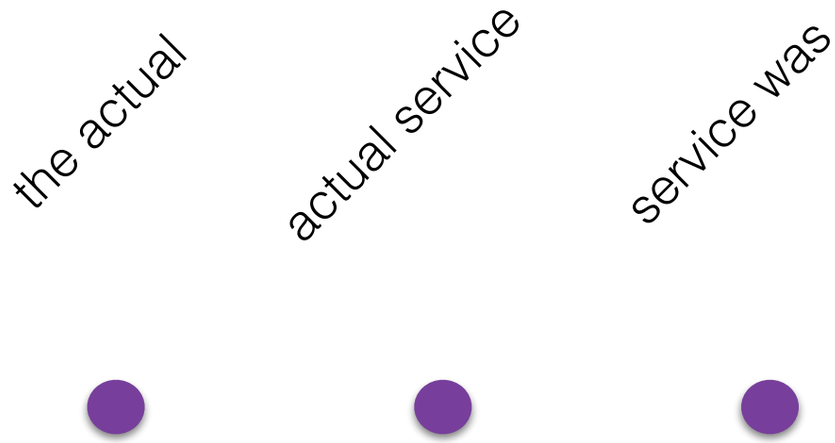
the actual

||

dot

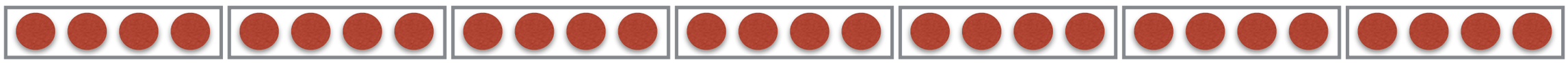
the actual service was not very good


the actual
actual service
dot
the
actual
service
was
not
very
good

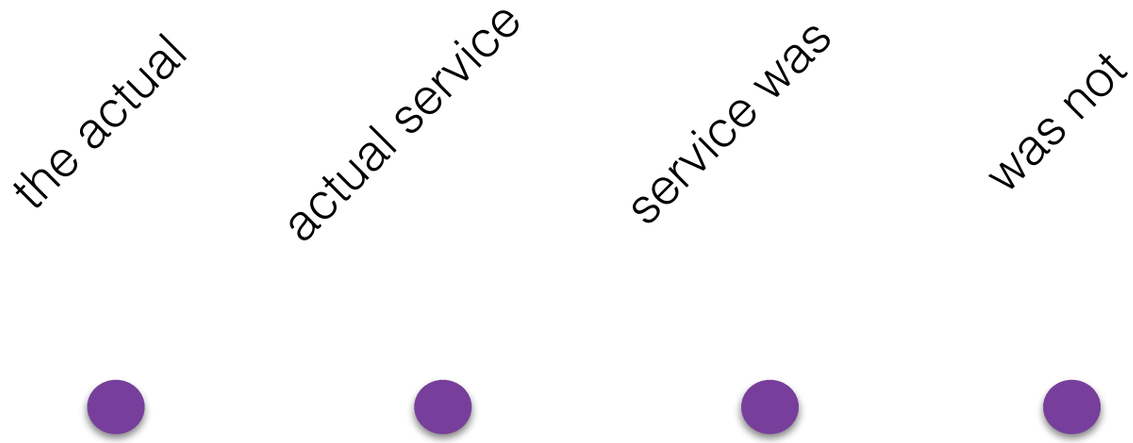


||

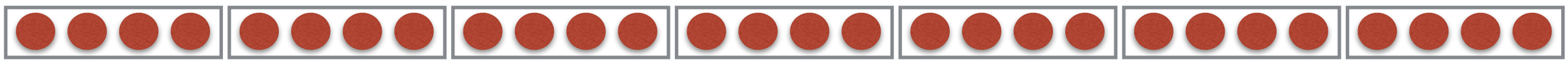
dot



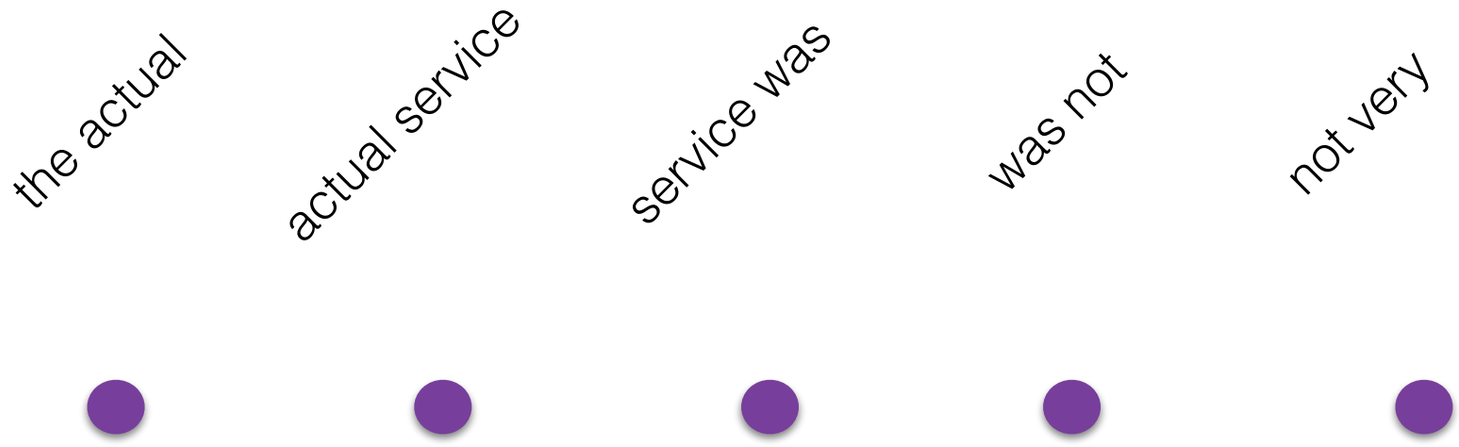
the actual service was not very good

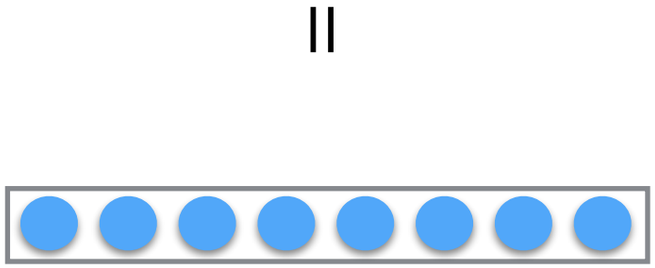


dot

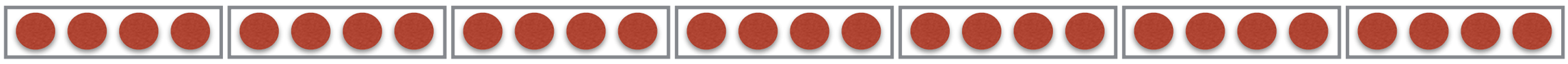


the actual service was not very good

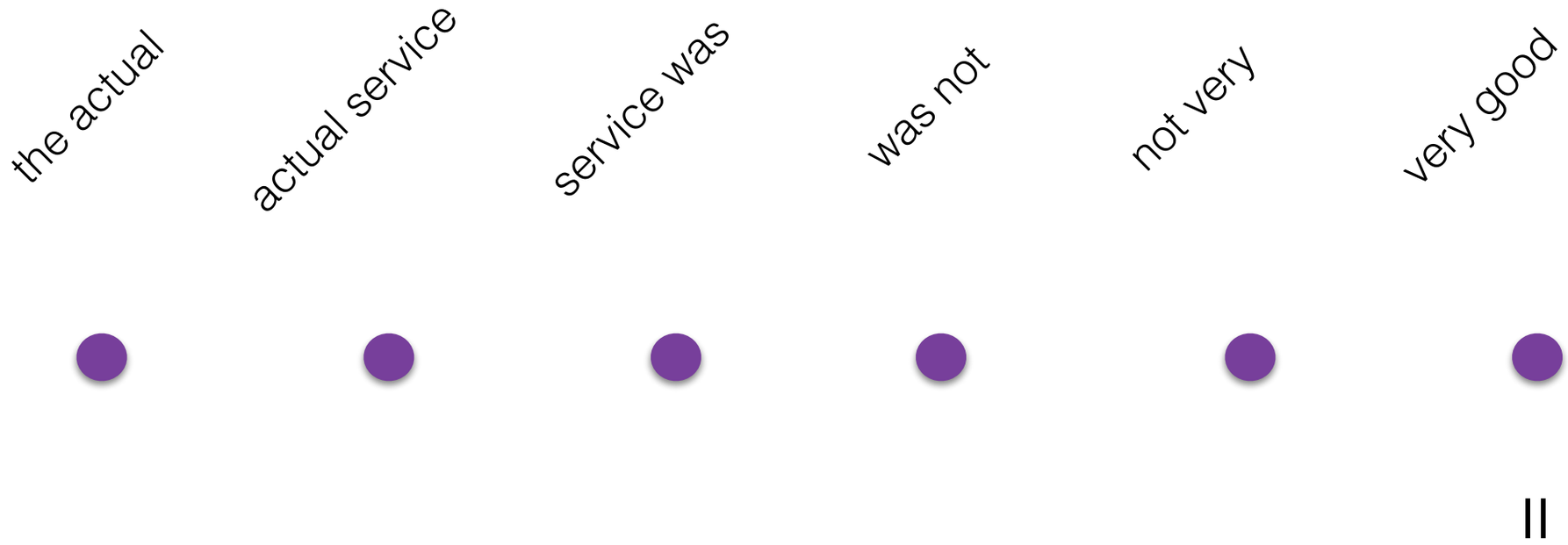




dot



the actual service was not very good


the actual
actual service
service was
was not
not very
very good
dot
the
actual
service
was
not
very
good

the actual

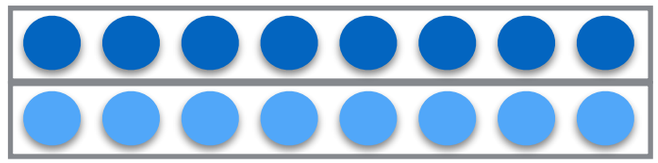
||

dot

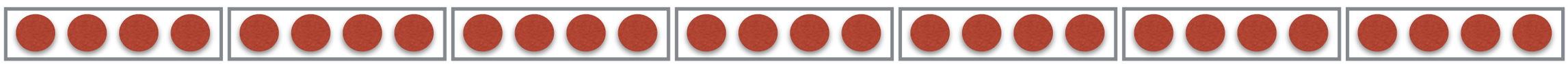
the actual service was not very good

the actual

||



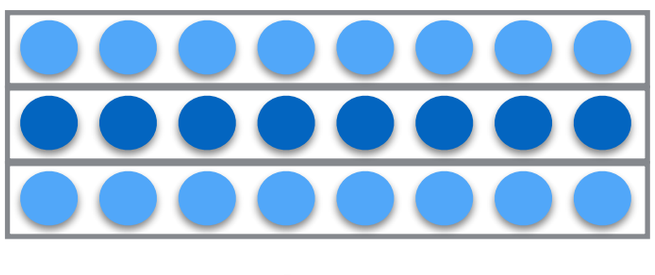
dot



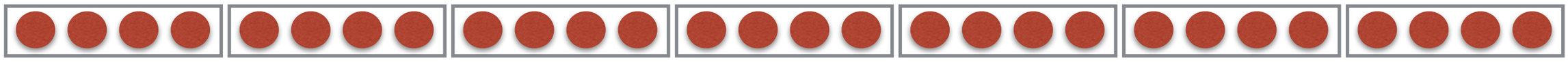
the actual service was not very good

the actual

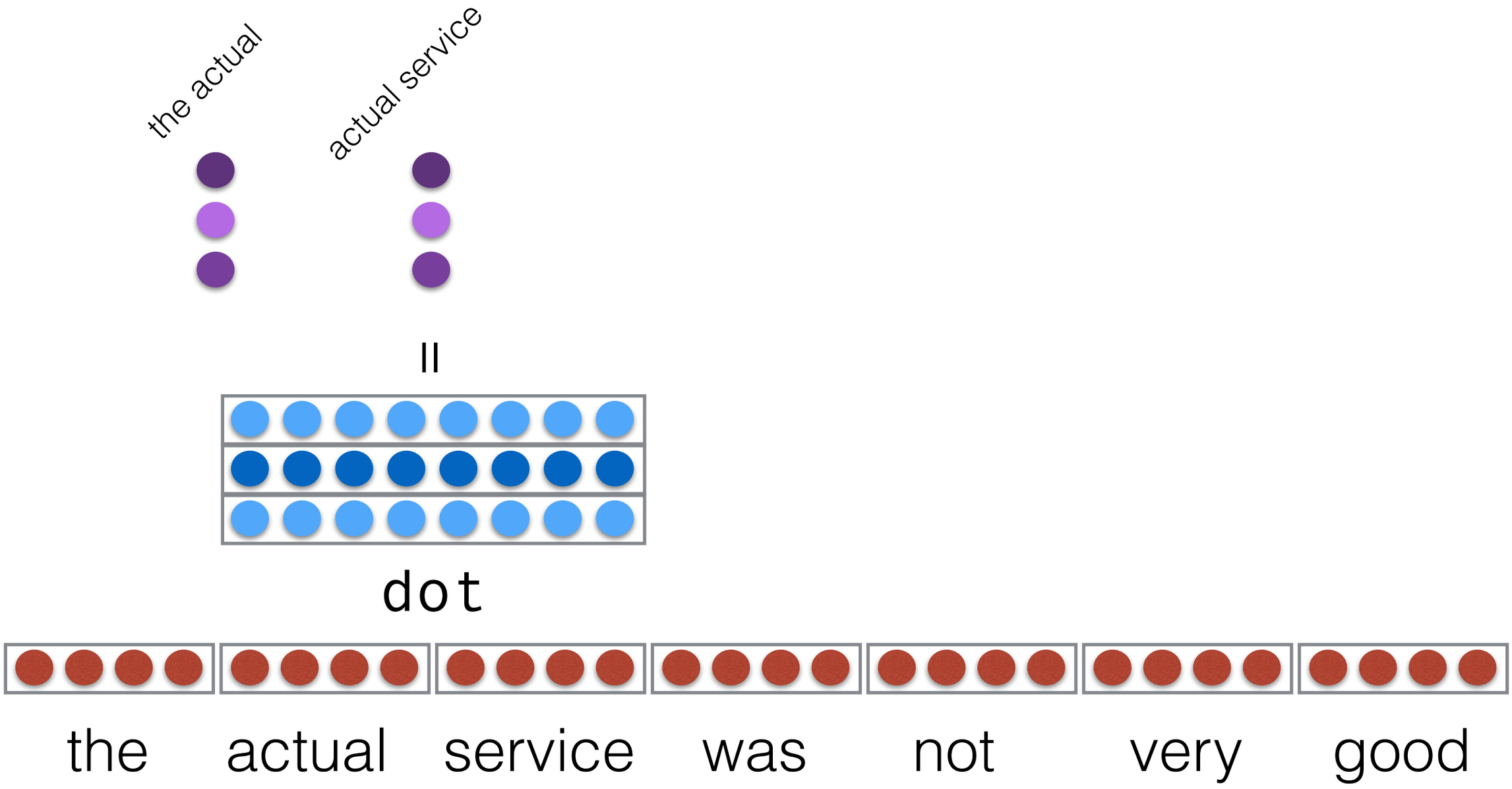
=

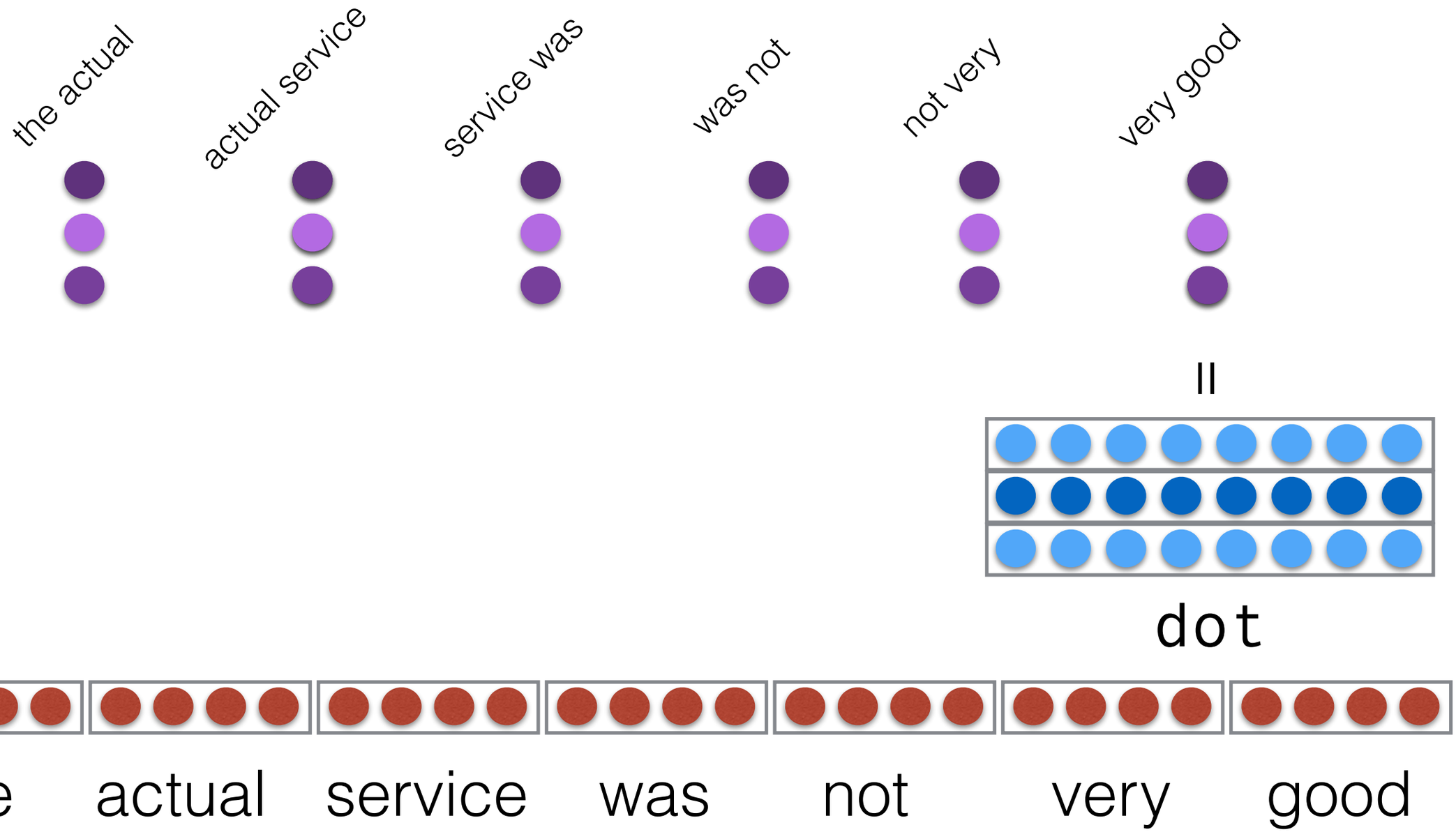


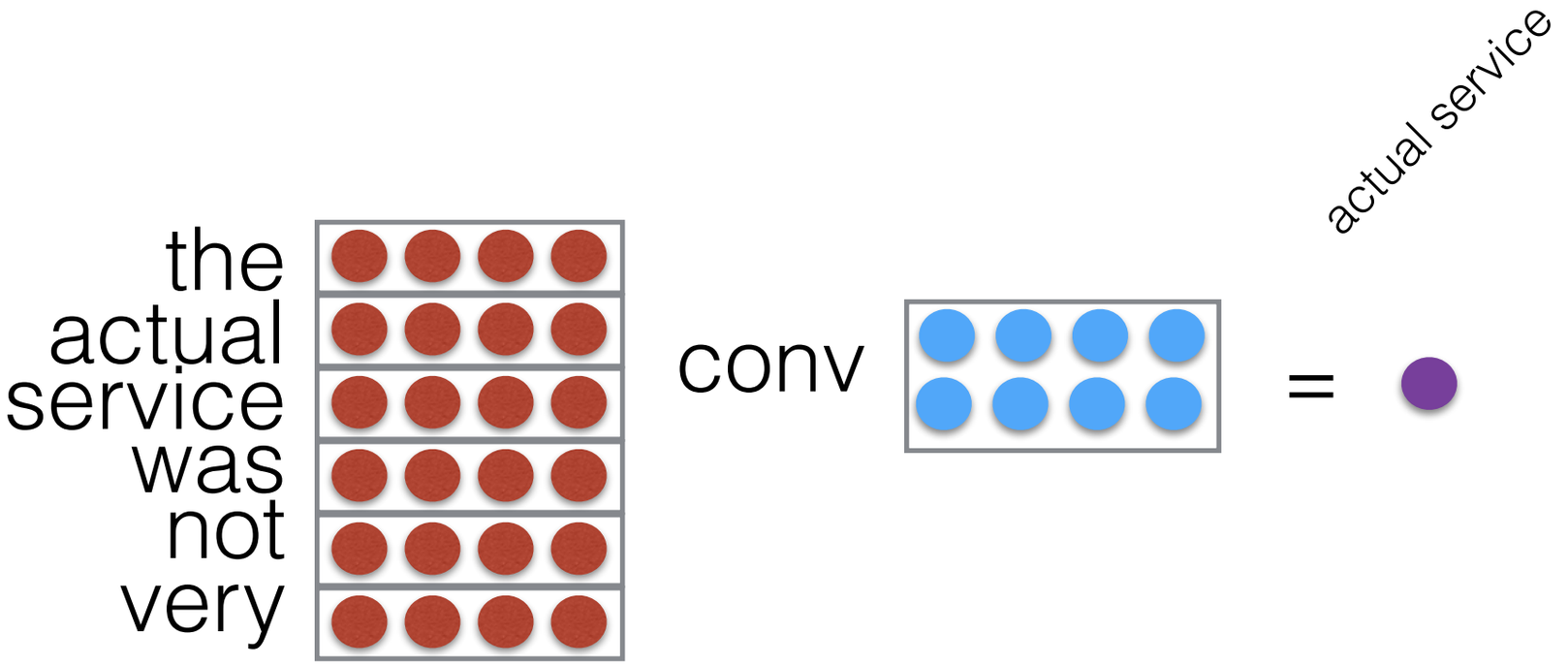
dot



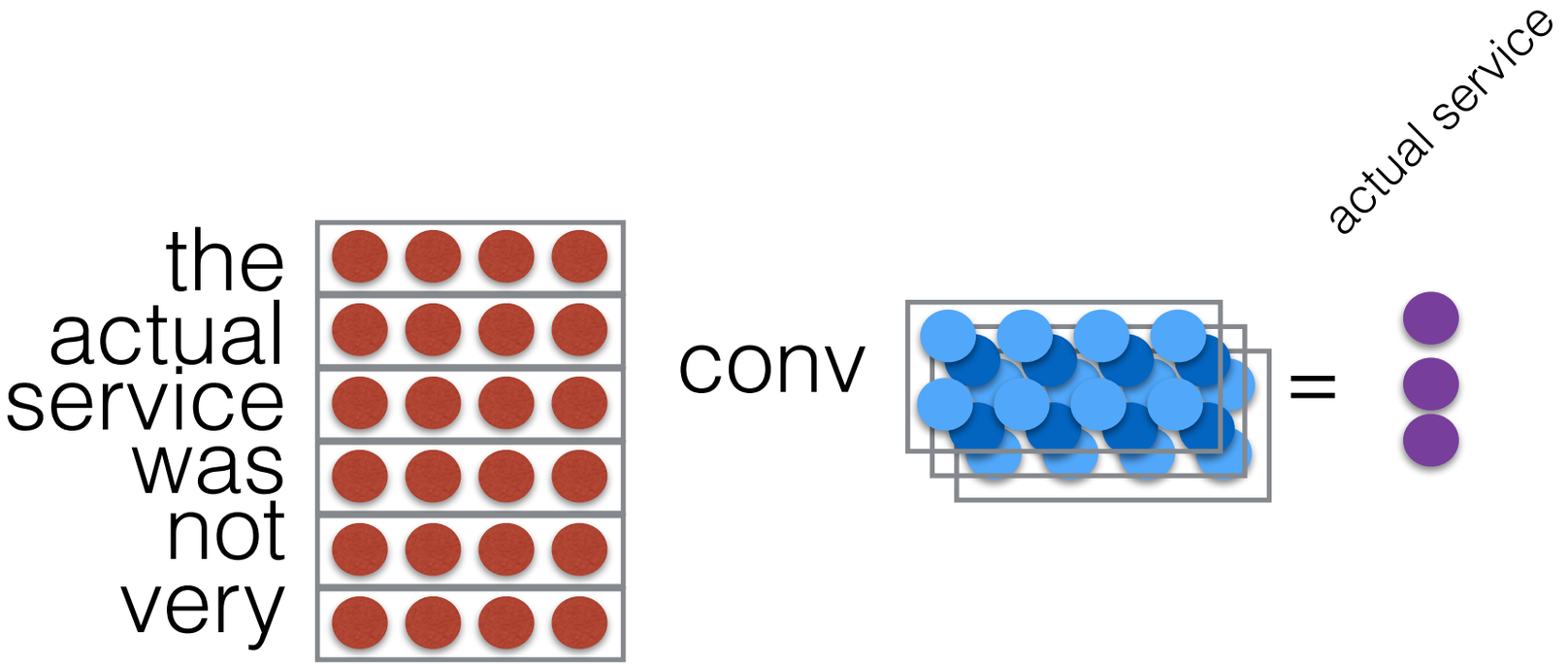
the actual service was not very good


the actual
actual service
=
dot
the
actual
service
was
not
very
good

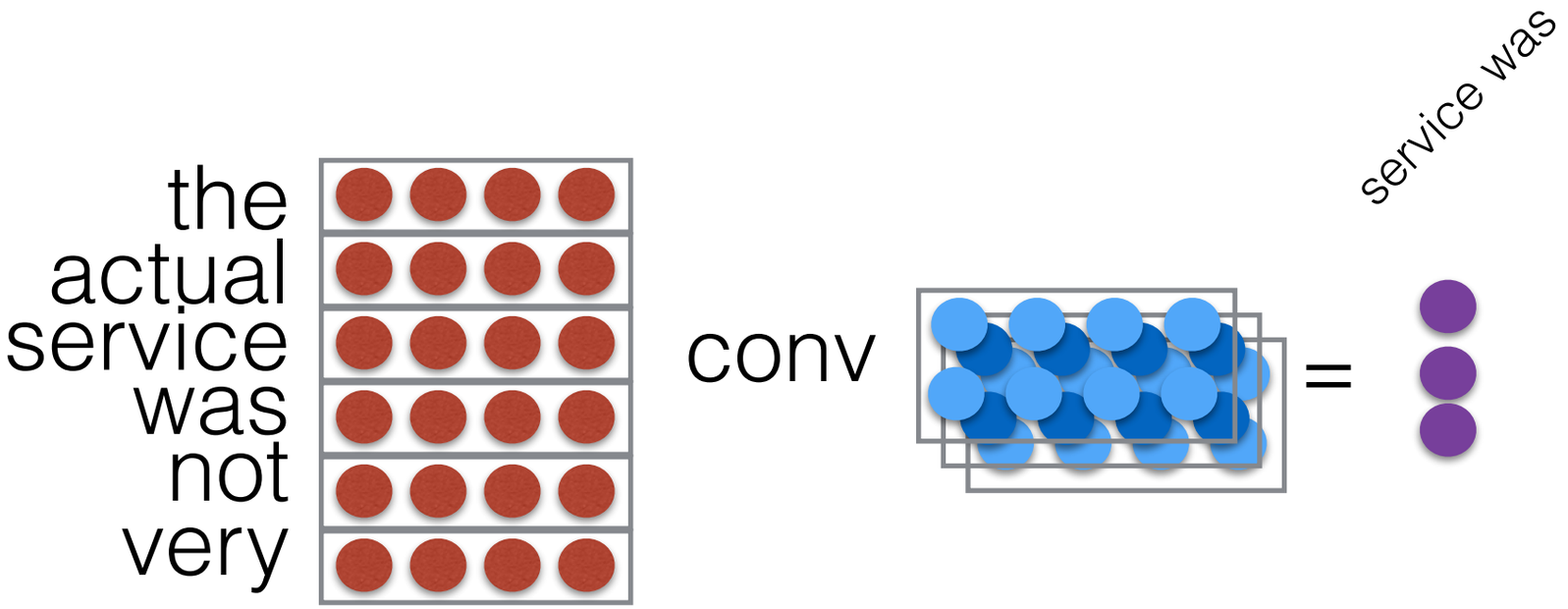

the actual
actual service
service was
was not
not very
very good
||
dot
the
actual
service
was
not
very
good



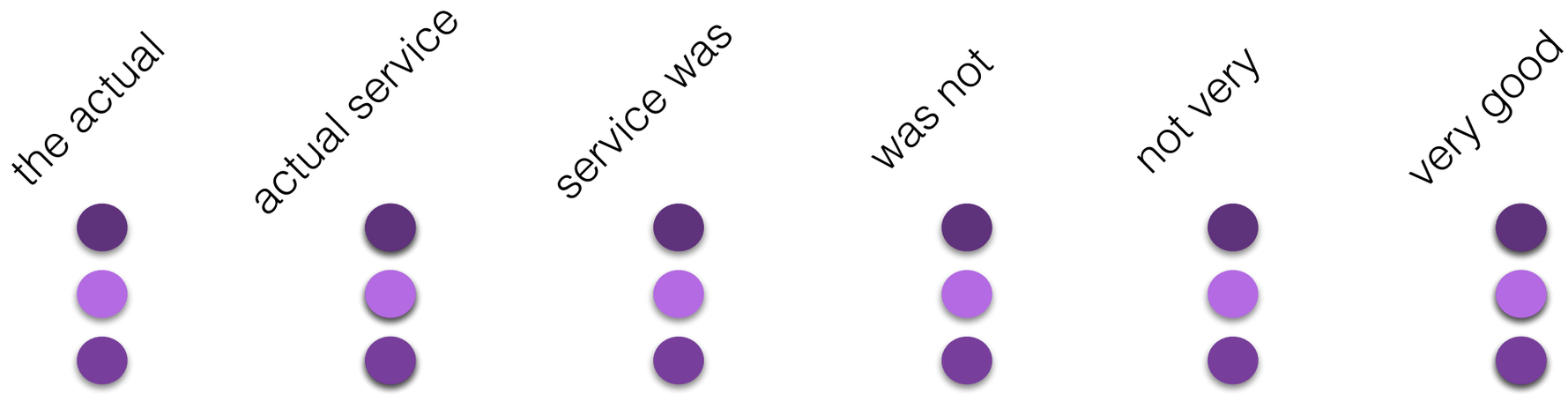
another way to represent text convolutions
(like 2d, useful for 2d-based software)

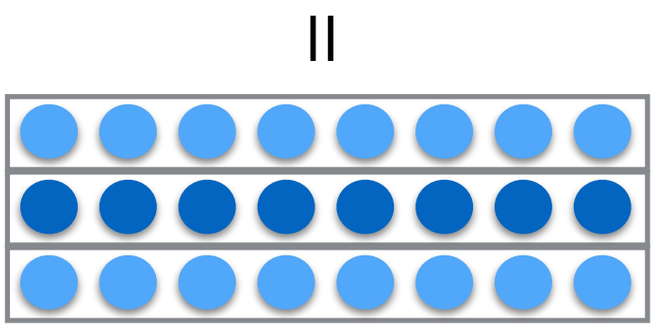


another way to represent text convolutions
(like 2d, useful for 2d-based software)

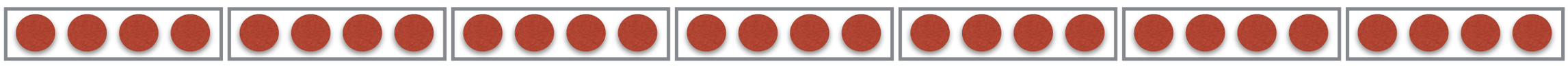


another way to represent text convolutions
(like 2d, useful for 2d-based software)



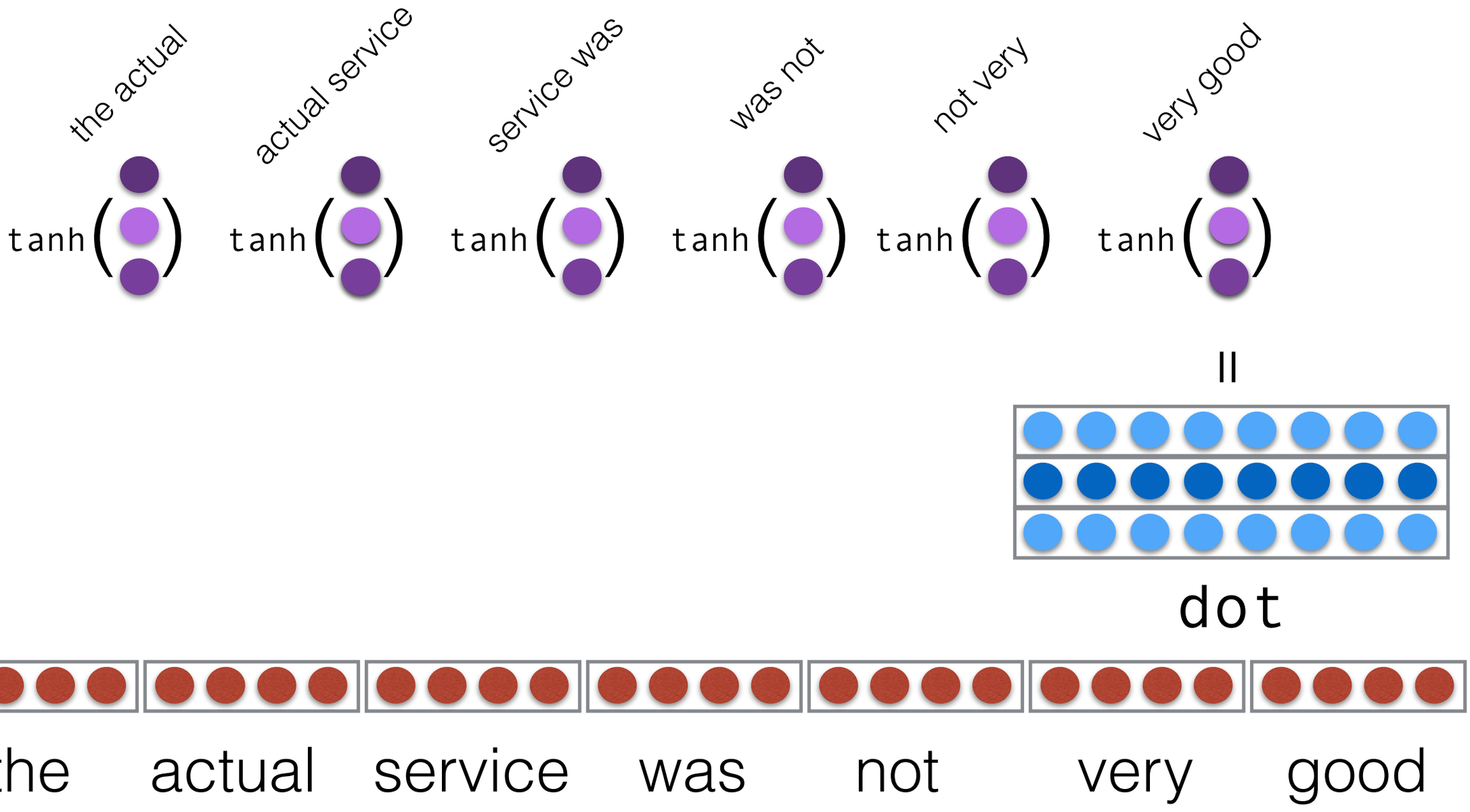


dot

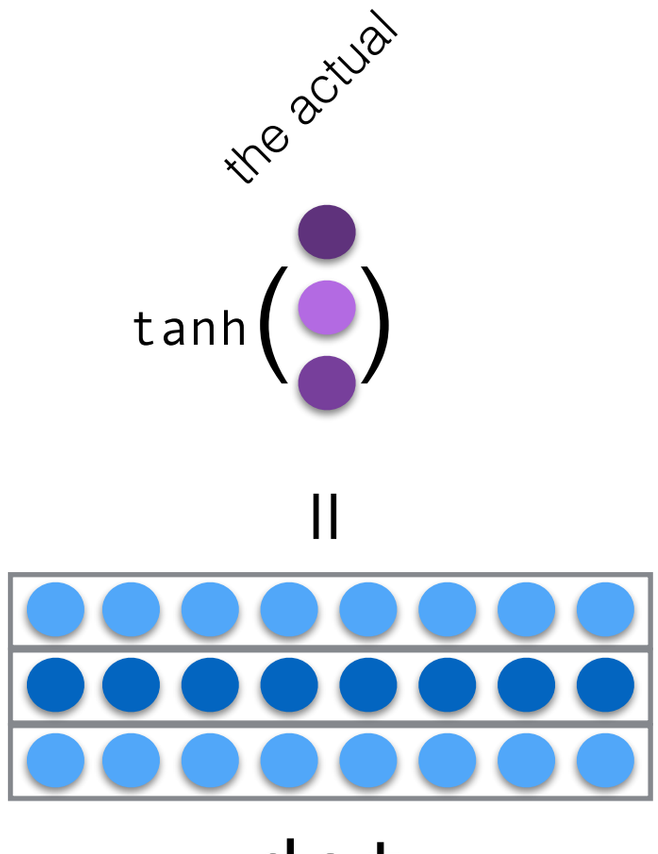


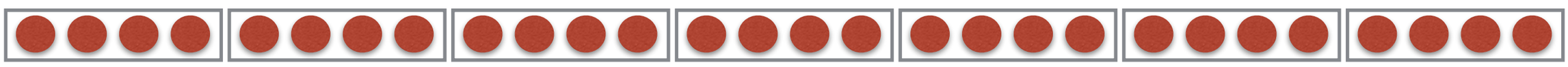
the actual service was not very good

**(we'll focus on the 1-d view here,
but remember they are equivalent)**



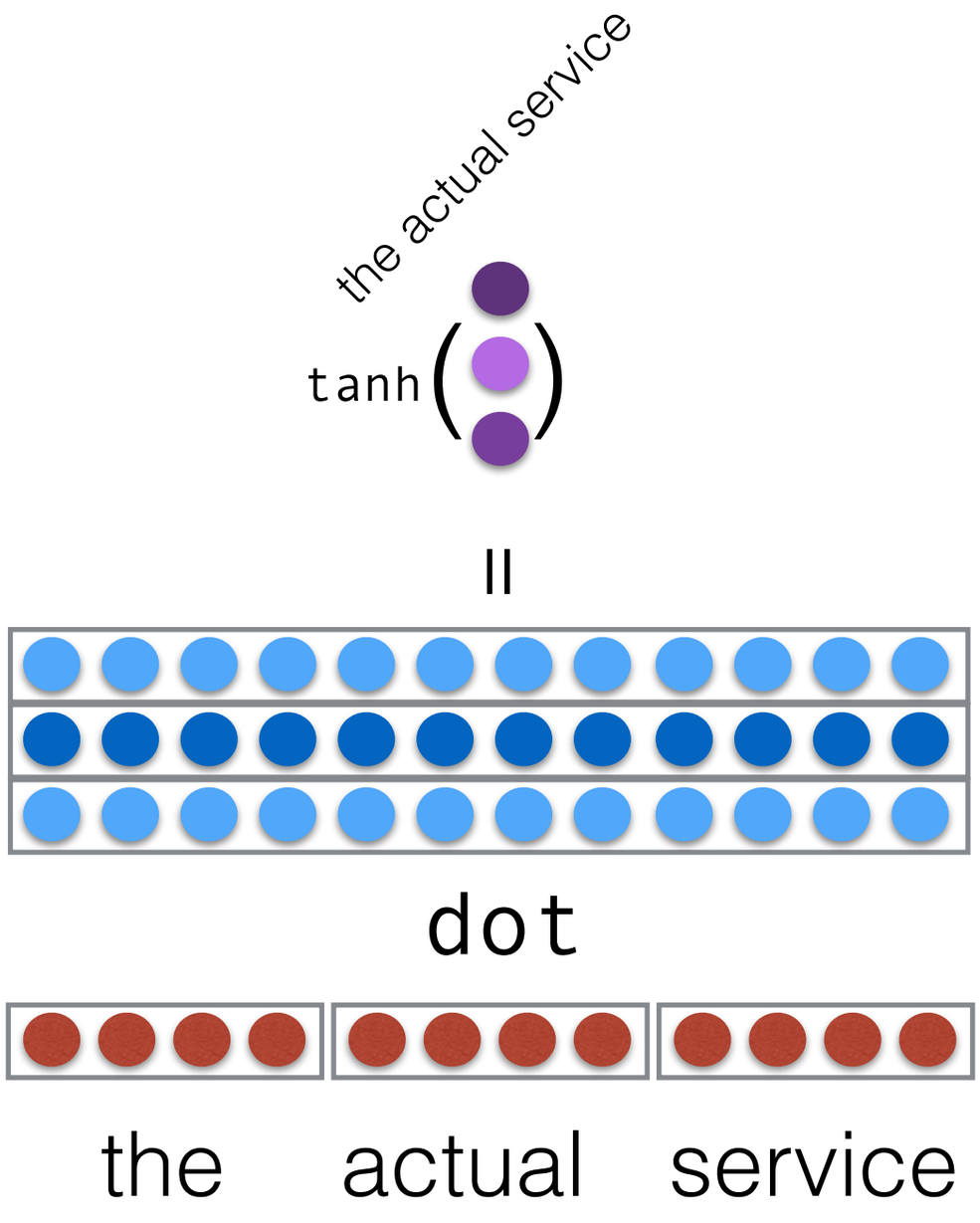
(usually also add non linearity)

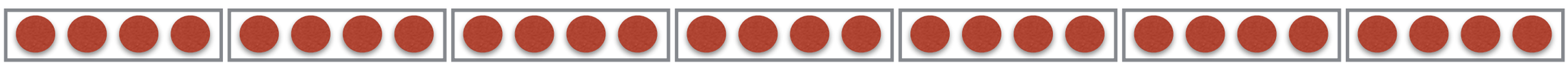




the actual service was not very good

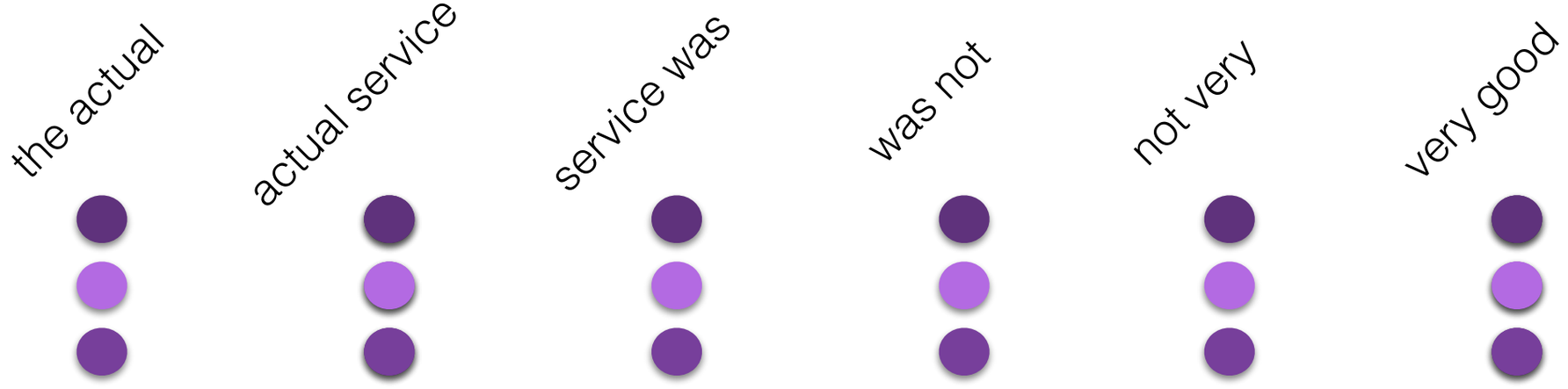
(can have larger filters...)

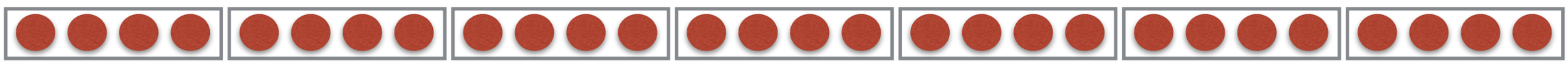




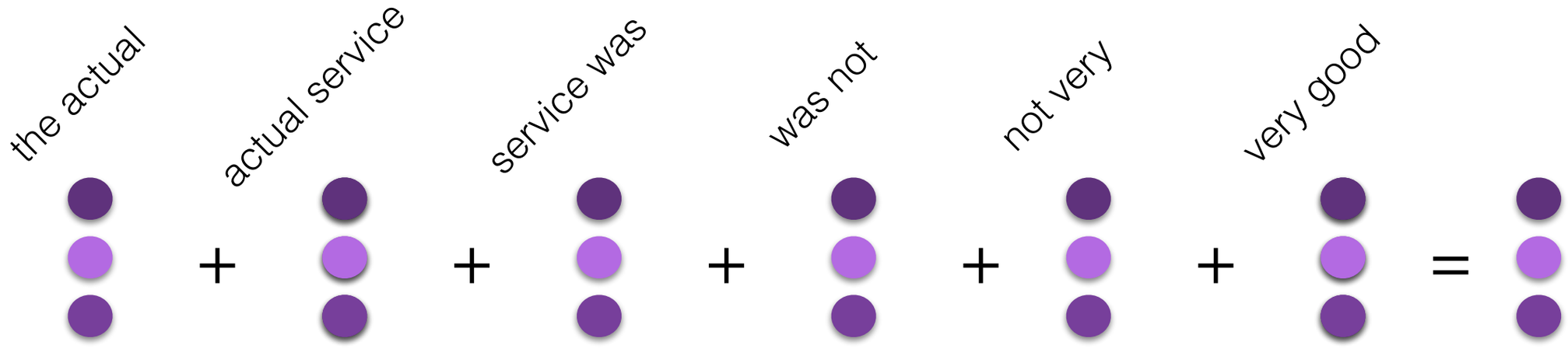
the actual service was not very good

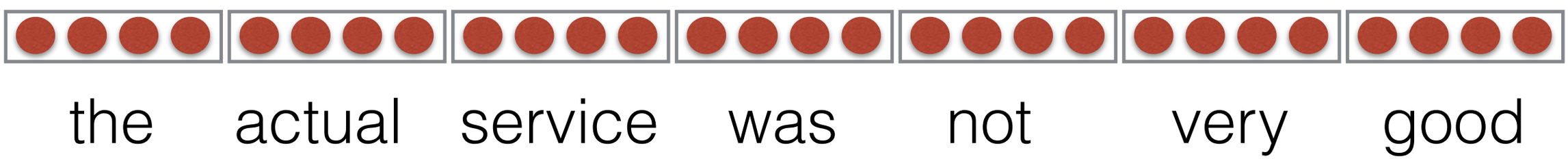
(...can have larger filters)





we have the ngram vectors. now what?





can do "pooling"

"Pooling"

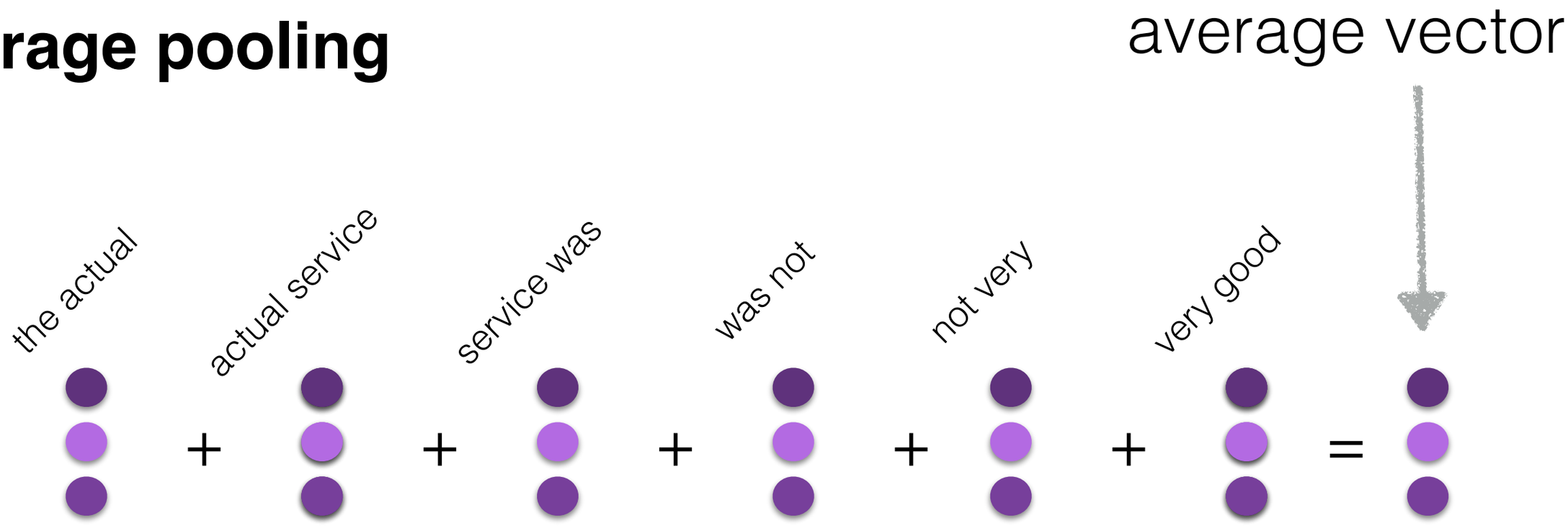
Combine K vectors into a single vector

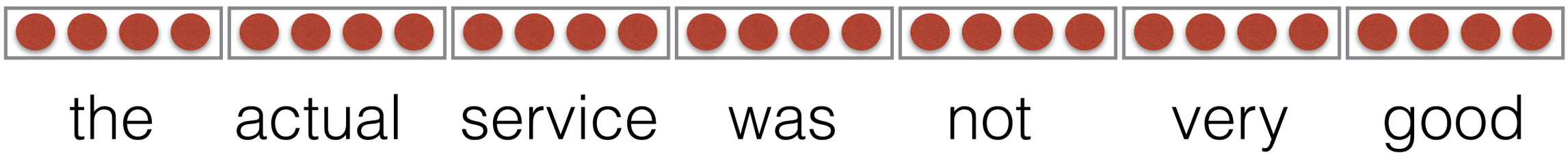
"Pooling"

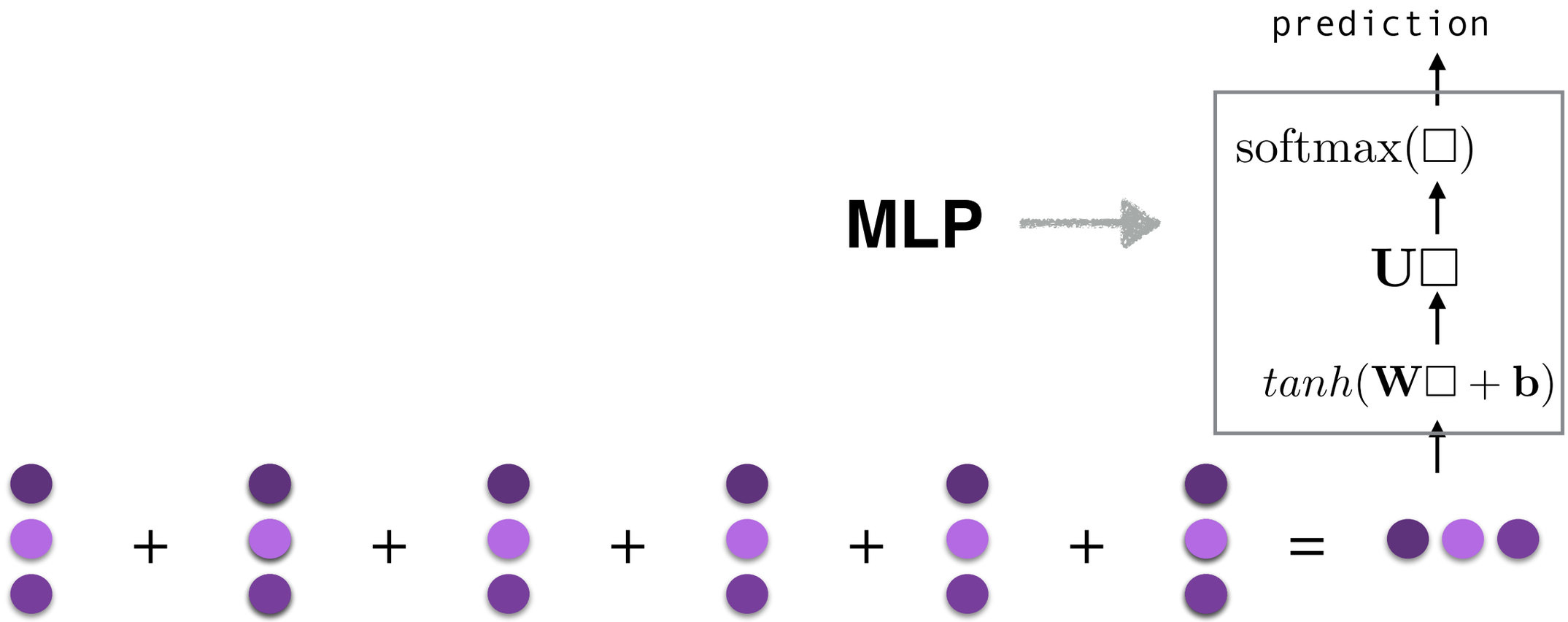
Combine K vectors into a single vector

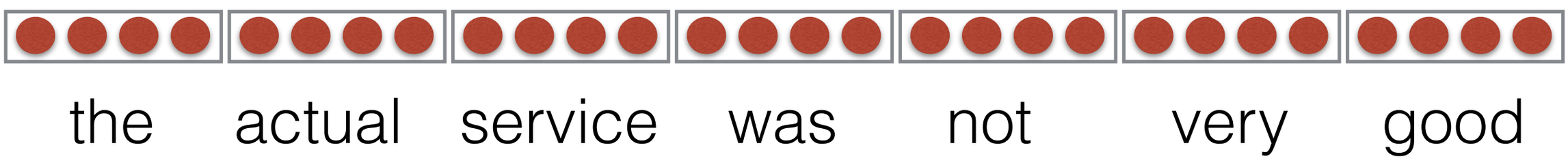
This vector is a summary of the K vectors, and can be used for prediction.

average pooling



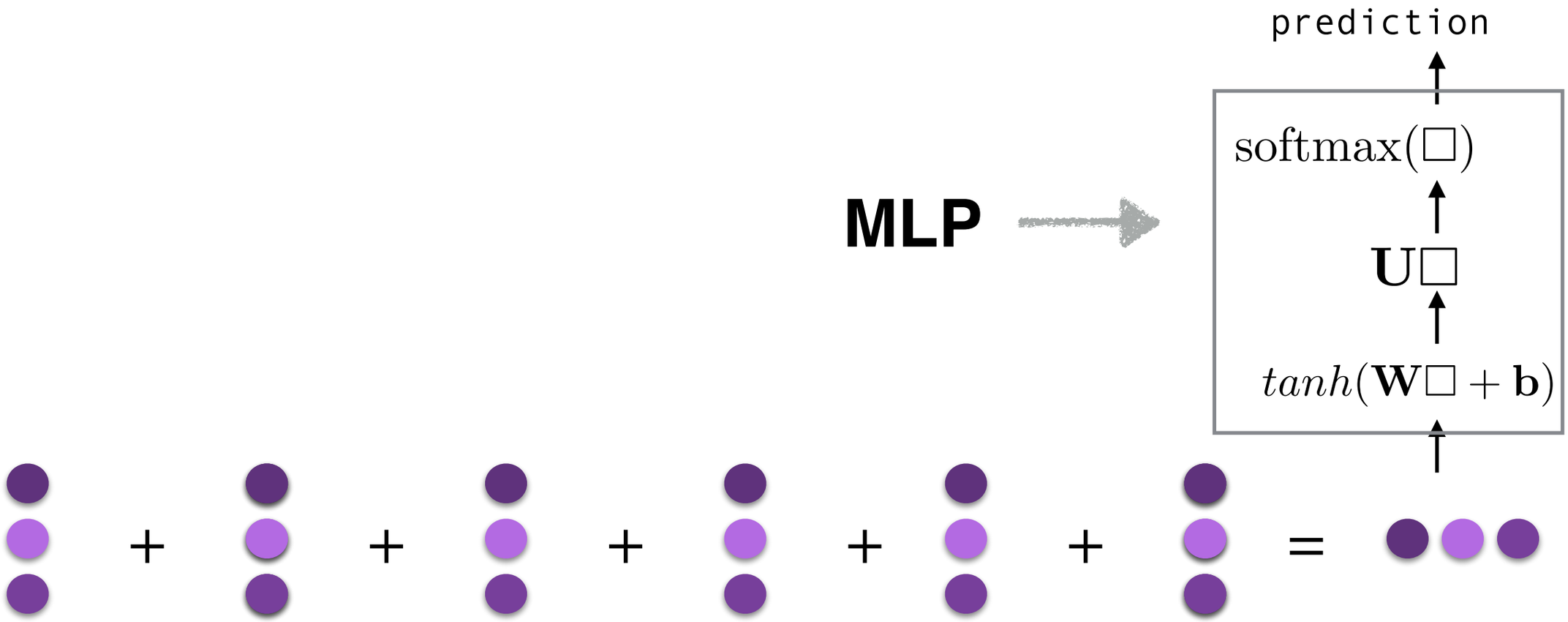


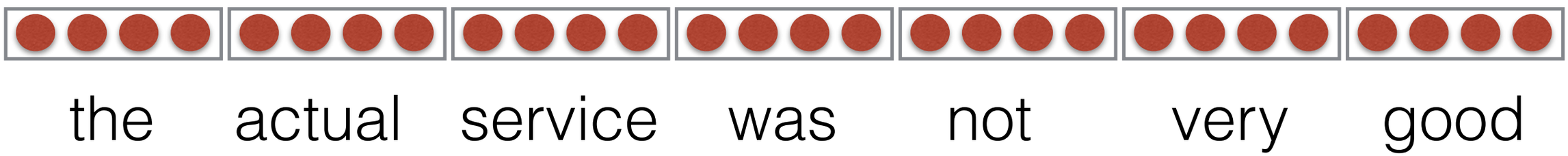




train end-to-end for some task

(train the MLP, the filter matrix, and the embeddings together)





train end-to-end for some task

(train the MLP, the filter matrix, and the embeddings together)

the vectors learn to capture what's important

we have the ngram vectors. now what?

Can look at the differences between terms.

microsoft *office* software		car *body* shop	
Free *office* 2000	0.550	car *body* kits	0.698
download *office* excel	0.541	auto *body* repair	0.578
word *office* online	0.502	auto *body* parts	0.555
apartment *office* hours	0.331	wave *body* language	0.301
massachusetts *office* location	0.293	calculate *body* fat	0.220
international *office* berkeley	0.274	forcefield *body* armour	0.165

Table 2: Sample word n-grams and the cosine similarities between the learned word-n-gram feature vectors of "*office*" and "*body*" in different contexts after the CLSM is trained.

A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval

Yelong Shen
Microsoft Research
Redmond, WA, USA
yeshen@microsoft.com

Xiaodong He
Microsoft Research
Redmond, WA, USA
xiaohe@microsoft.com

Jianfeng Gao
Microsoft Research
Redmond, WA, USA
jfgao@microsoft.com

Li Deng
Microsoft Research
Redmond, WA, USA
deng@microsoft.com

Grégoire Mesnil
University of Montréal
Montréal, Canada
gregoire.mesnil@umontreal.ca

we have the ngram vectors. now what?

Can look at the differences between terms.

microsoft *office* software		car *body* shop	
Free *office* 2000	0.550	car *body* kits	0.698
download *office* excel	0.541	auto *body* repair	0.578
word *office* online	0.502	auto *body* parts	0.555
apartment *office* hours	0.331	wave *body* language	0.301
massachusetts *office* location	0.293	calculate *body* fat	0.220
international *office* berkeley	0.274	forcefield *body* armour	0.165

Table 2: Sample word n-grams and the cosine similarities between the learned word-n-gram feature vectors of "*office*" and "*body*" in different contexts after the CLSM is trained.

A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval

Yelong Shen
Microsoft Research
Redmond, WA, USA
yeshen@microsoft.com

Xiaodong He
Microsoft Research
Redmond, WA, USA
xiaohe@microsoft.com

Jianfeng Gao
Microsoft Research
Redmond, WA, USA
jfgao@microsoft.com

Li Deng
Microsoft Research
Redmond, WA, USA
deng@microsoft.com

Grégoire Mesnil
University of Montréal
Montréal, Canada
gregoire.mesnil@umontreal.ca

we have the ngram vectors. now what?

Can look at the differences between terms.

microsoft *office* software		car *body* shop	
Free *office* 2000	0.550	car *body* kits	0.698
download *office* excel	0.541	auto *body* repair	0.578
word *office* online	0.502	auto *body* parts	0.555
apartment *office* hours	0.331	wave *body* language	0.301
massachusetts *office* location	0.293	calculate *body* fat	0.220
international *office* berkeley	0.274	forcefield *body* armour	0.165

Table 2: Sample word n-grams and the cosine similarities between the learned word-n-gram feature vectors of "*office*" and "*body*" in different contexts after the CLSM is trained.

A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval

Yelong Shen
Microsoft Research
Redmond, WA, USA
yeshen@microsoft.com

Xiaodong He
Microsoft Research
Redmond, WA, USA
xiaohe@microsoft.com

Jianfeng Gao
Microsoft Research
Redmond, WA, USA
jfgao@microsoft.com

Li Deng
Microsoft Research
Redmond, WA, USA
deng@microsoft.com

Grégoire Mesnil
University of Montréal
Montréal, Canada
gregoire.mesnil@umontreal.ca

we have the ngram vectors. now what?

Can look at the differences between terms.

microsoft *office* software		car *body* shop	
Free *office* 2000	0.550	car *body* kits	0.698
download *office* excel	0.541	auto *body* repair	0.578
word *office* online	0.502	auto *body* parts	0.555
apartment *office* hours	0.331	wave *body* language	0.301
massachusetts *office* location	0.293	calculate *body* fat	0.220
international *office* berkeley	0.274	forcefield *body* armour	0.165

Table 2: Sample word n-grams and the cosine similarities between the learned word-n-gram feature vectors of "*office*" and "*body*" in different contexts after the CLSM is trained.

A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval

Yelong Shen
Microsoft Research
Redmond, WA, USA
yeshen@microsoft.com

Xiaodong He
Microsoft Research
Redmond, WA, USA
xiaohe@microsoft.com

Jianfeng Gao
Microsoft Research
Redmond, WA, USA
jfgao@microsoft.com

Li Deng
Microsoft Research
Redmond, WA, USA
deng@microsoft.com

Grégoire Mesnil
University of Montréal
Montréal, Canada
gregoire.mesnil@umontreal.ca

we have the ngram vectors. now what?

Can look at the differences between terms.

microsoft *office* software		car *body* shop	
Free *office* 2000	0.550	car *body* kits	0.698
download *office* excel	0.541	auto *body* repair	0.578
word *office* online	0.502	auto *body* parts	0.555
apartment *office* hours	0.331	wave *body* language	0.301
massachusetts *office* location	0.293	calculate *body* fat	0.220
international *office* berkeley	0.274	forcefield *body* armour	0.165

Table 2: Sample word n-grams and the cosine similarities between the learned word-n-gram feature vectors of "*office*" and "*body*" in different contexts after the CLSM is trained.

A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval

Yelong Shen
Microsoft Research
Redmond, WA, USA
yeshen@microsoft.com

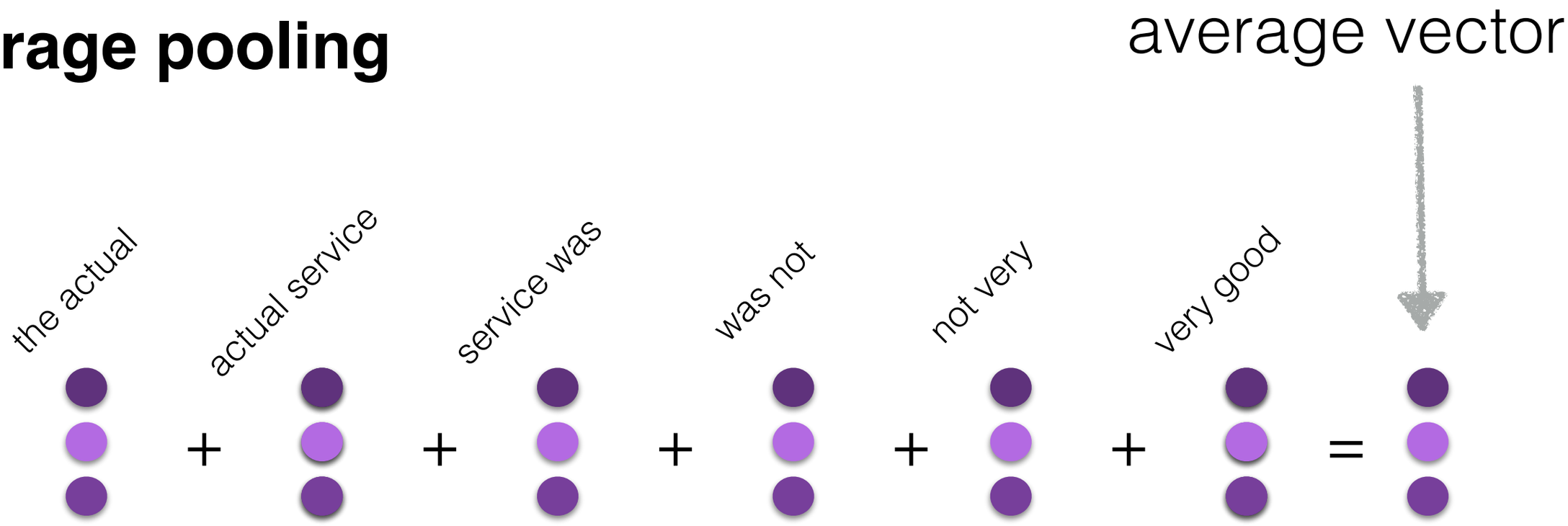
Xiaodong He
Microsoft Research
Redmond, WA, USA
xiaohe@microsoft.com

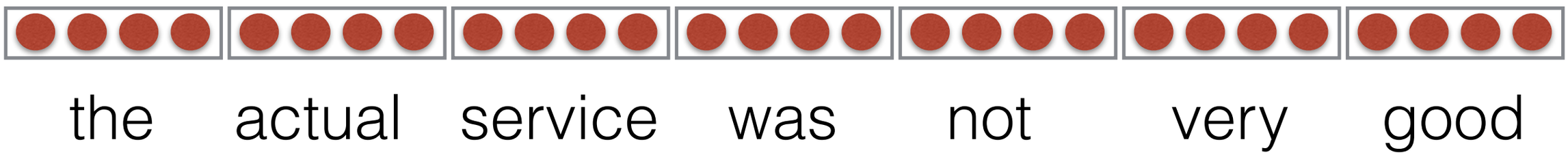
Jianfeng Gao
Microsoft Research
Redmond, WA, USA
jfgao@microsoft.com

Li Deng
Microsoft Research
Redmond, WA, USA
deng@microsoft.com

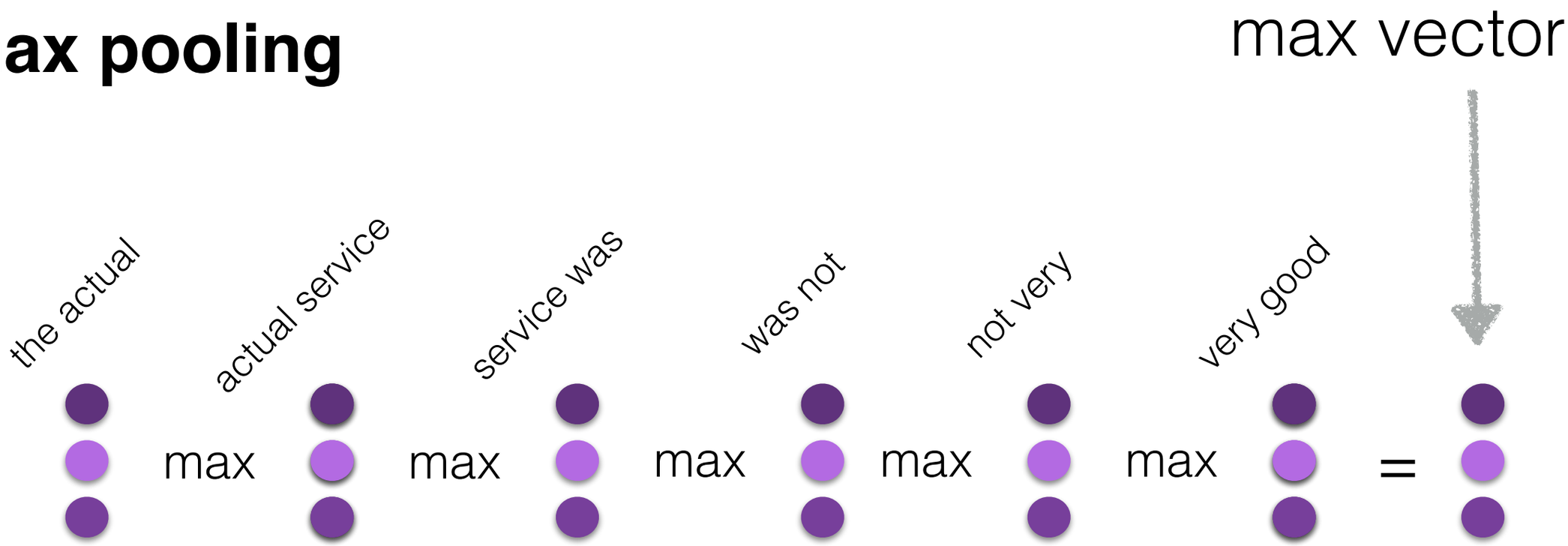
Grégoire Mesnil
University of Montréal
Montréal, Canada
gregoire.mesnil@umontreal.ca

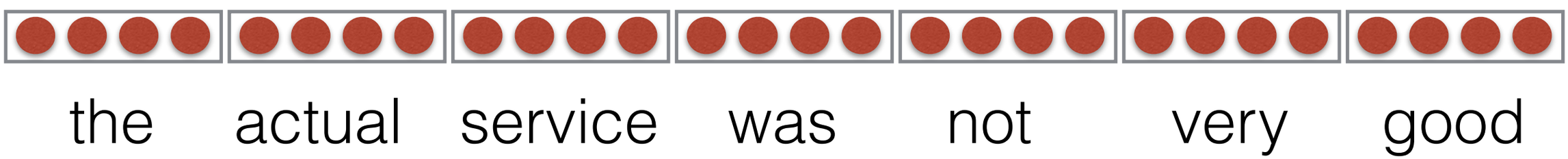
average pooling





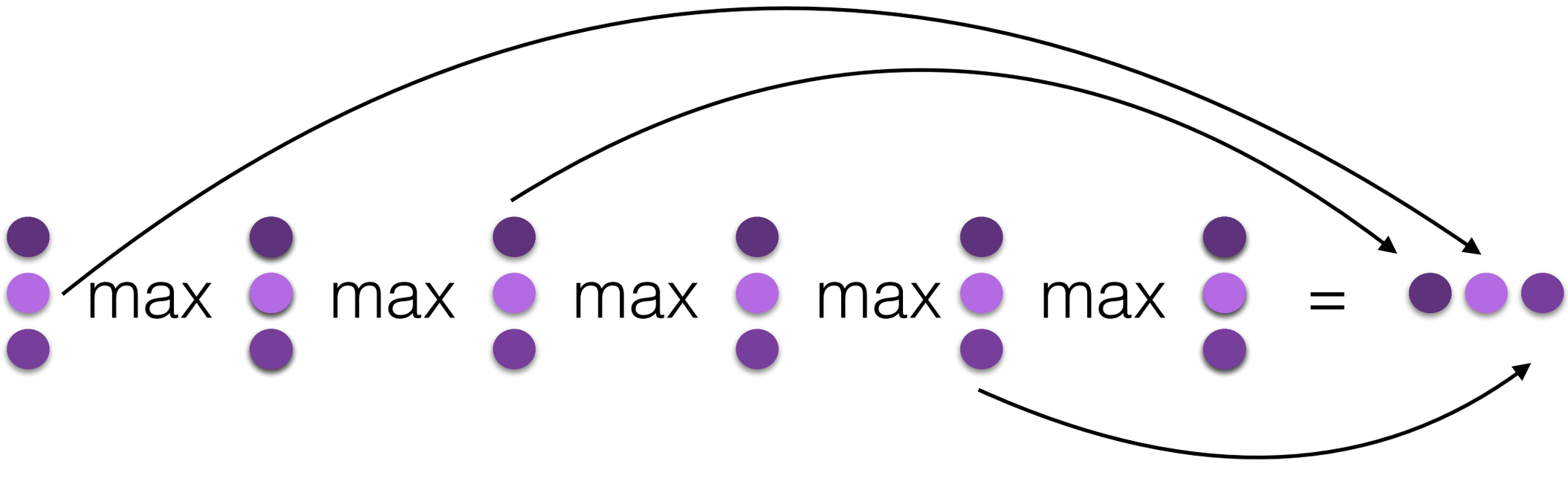
max pooling

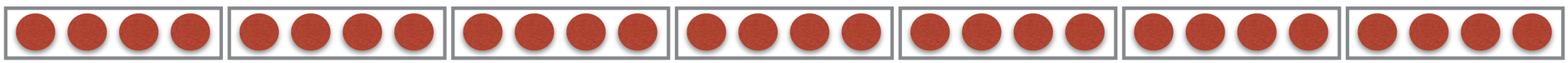




(max in each coordinate)

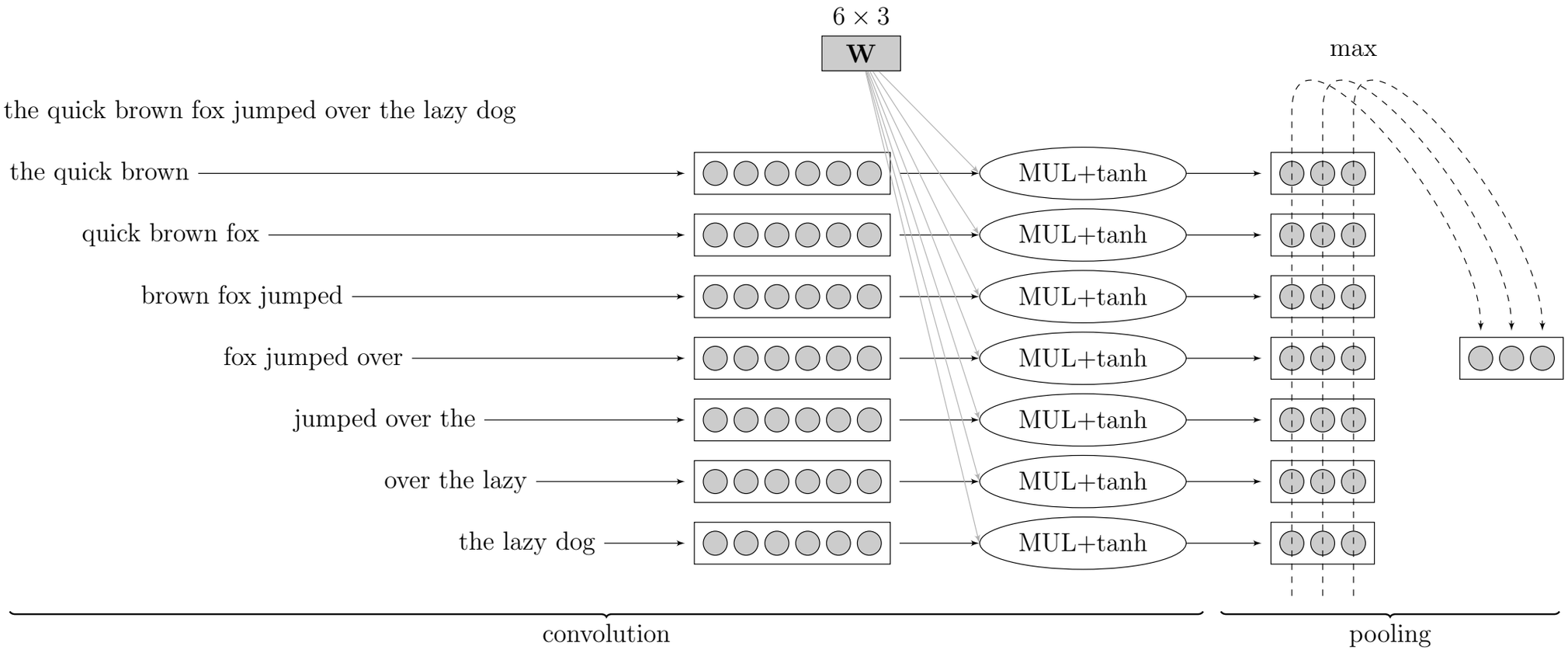
each dimension comes from a specific ngram



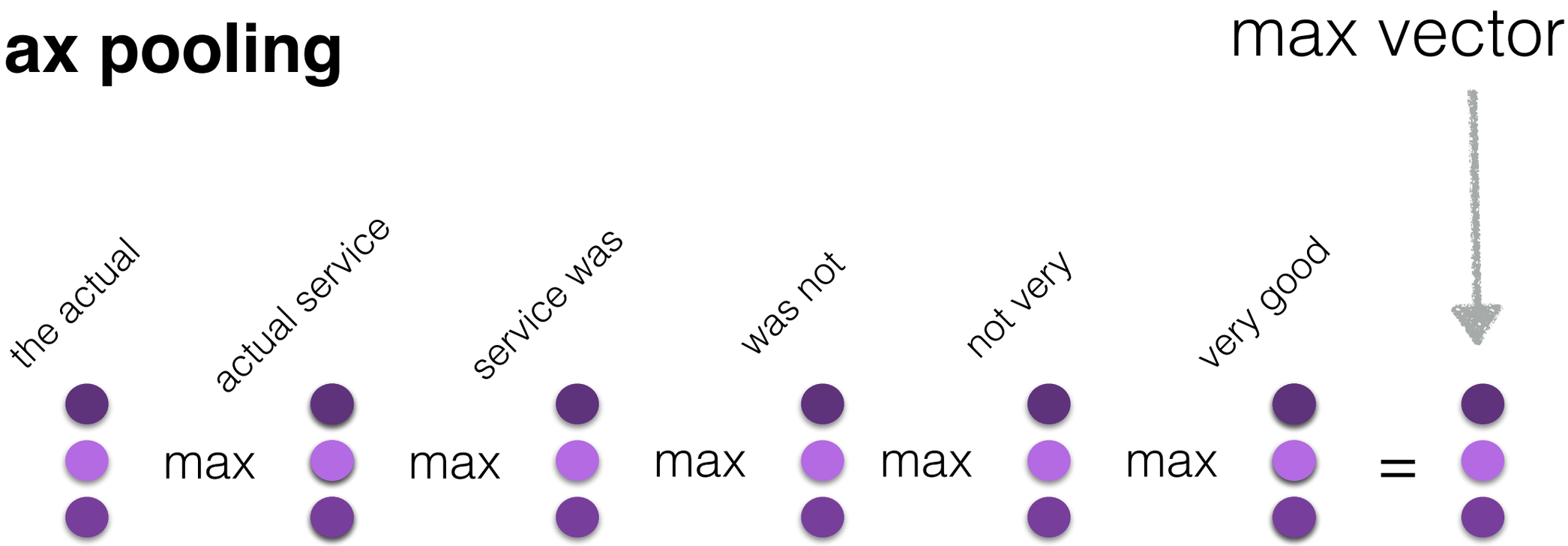


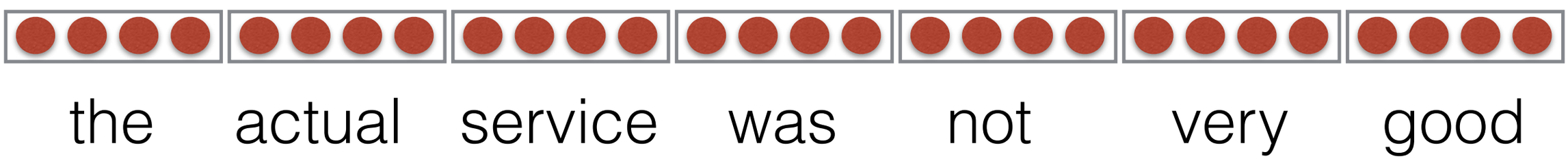
the actual service was not very good

Another way to draw this:



max pooling



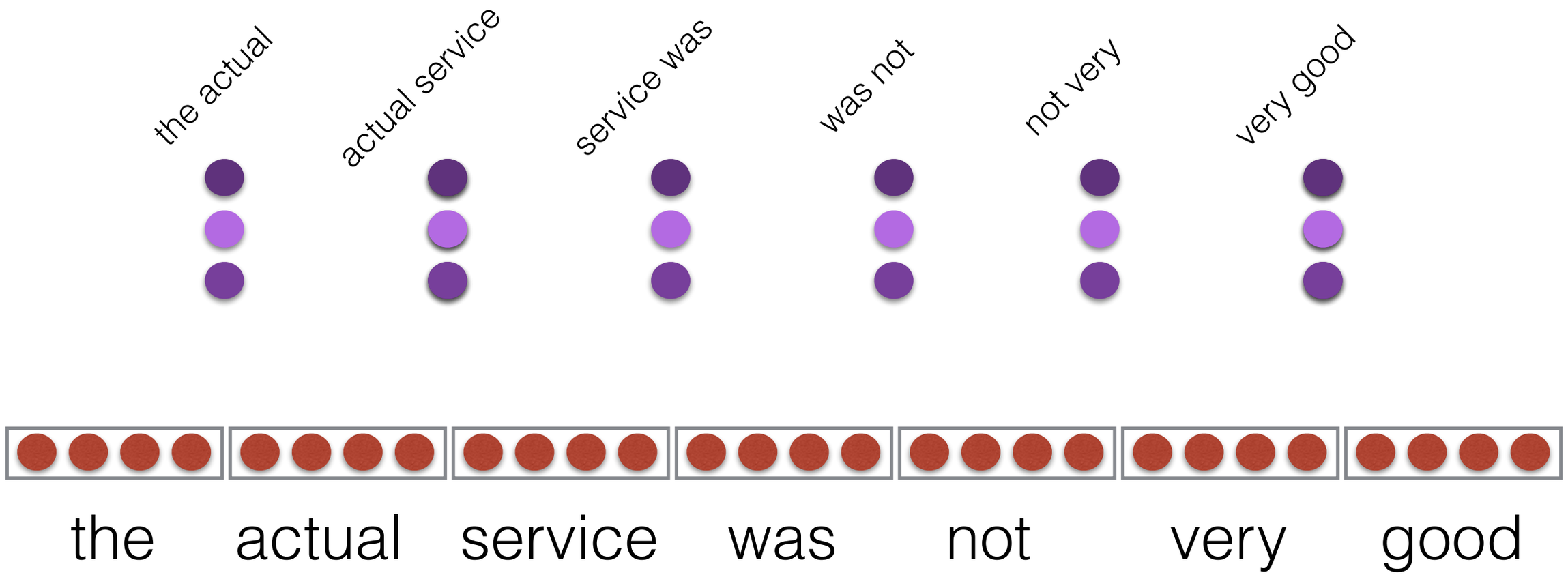


max vs average -- discuss

one benefit of max-pooling: it's "**interpretable**"

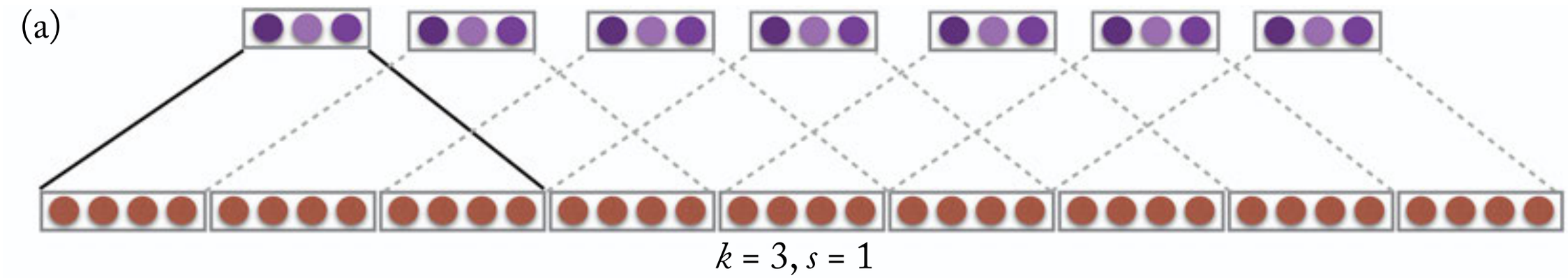
we can know where each element
in the summary vector came from

Strides



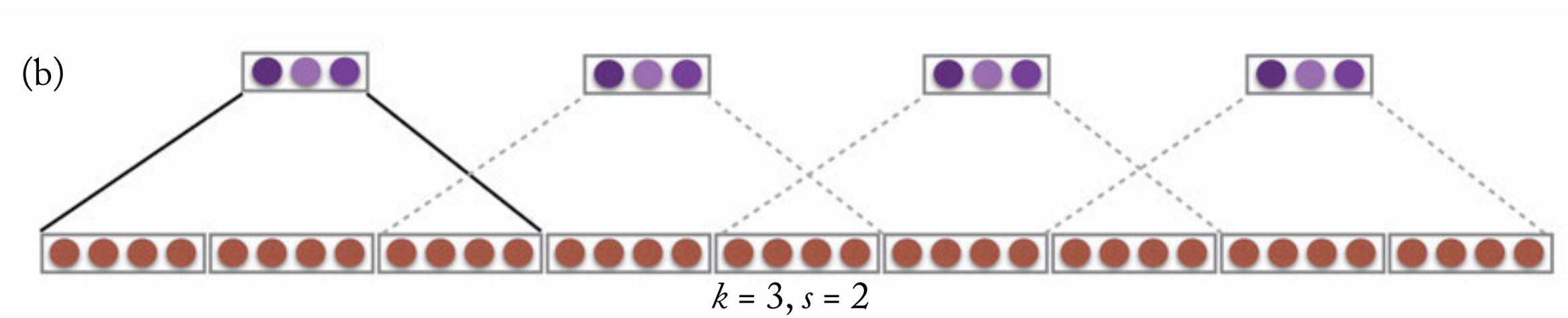
strides = how much you move

Strides



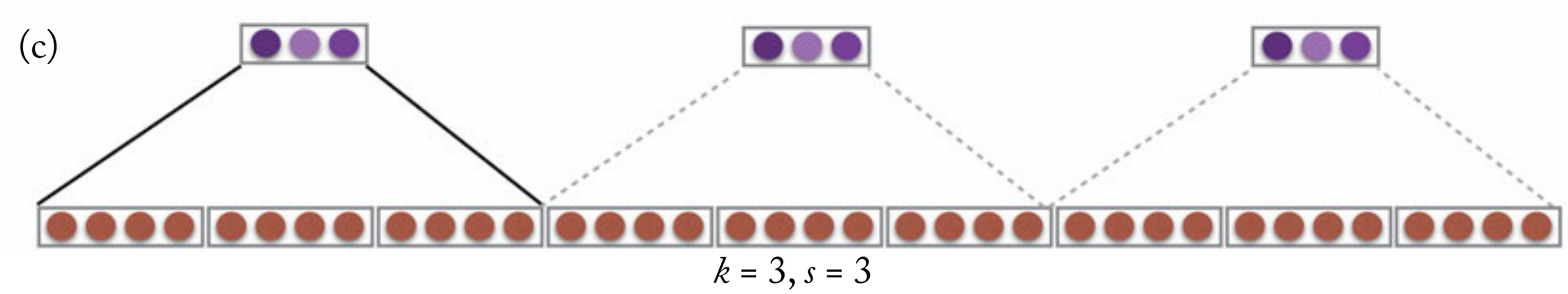
k = 3, stride = 1

Strides



k = 3, stride = 2

Strides



k = 3, stride = 3

In the world

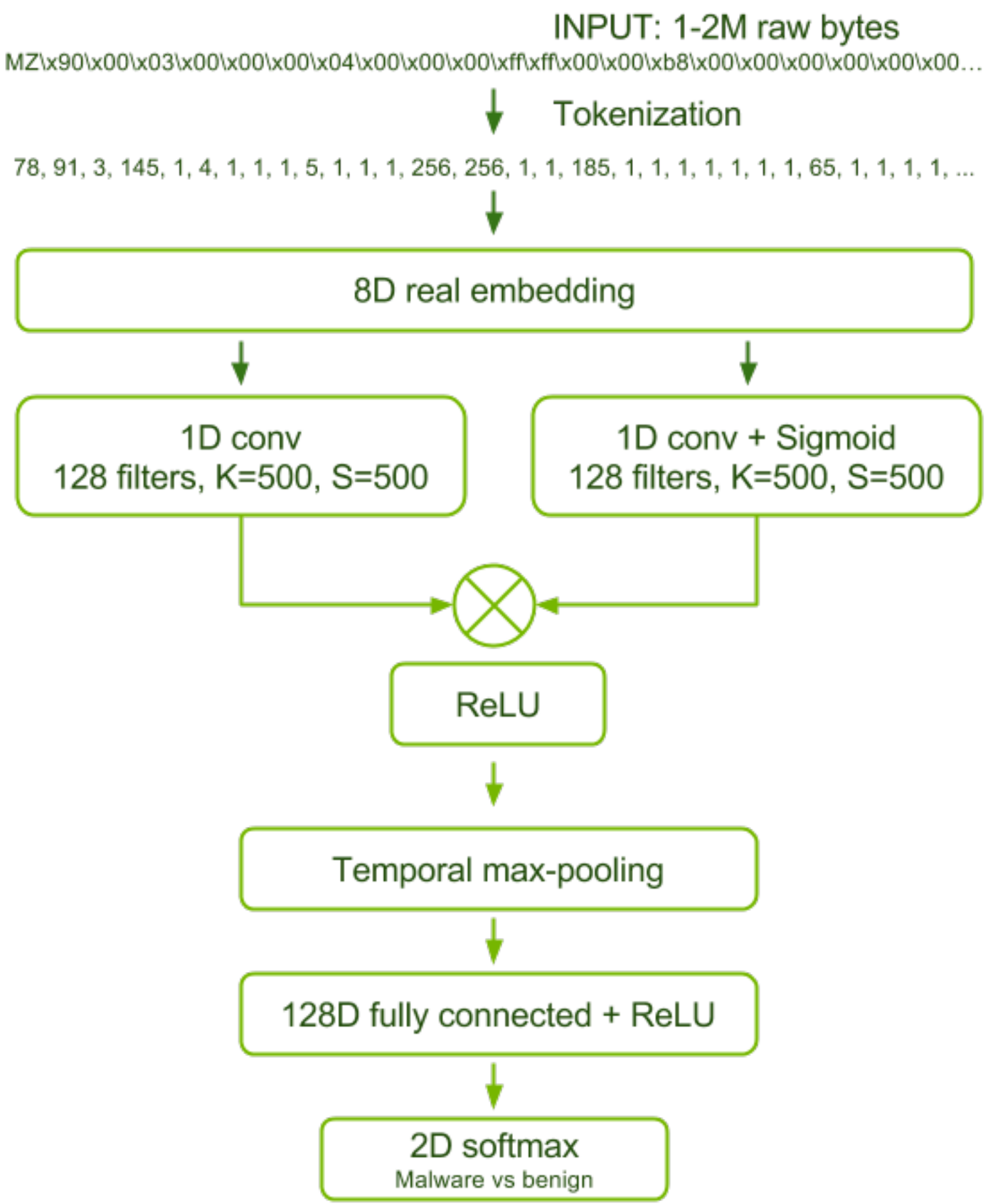
Malware Detection in Executables Using Neural Networks

Share:

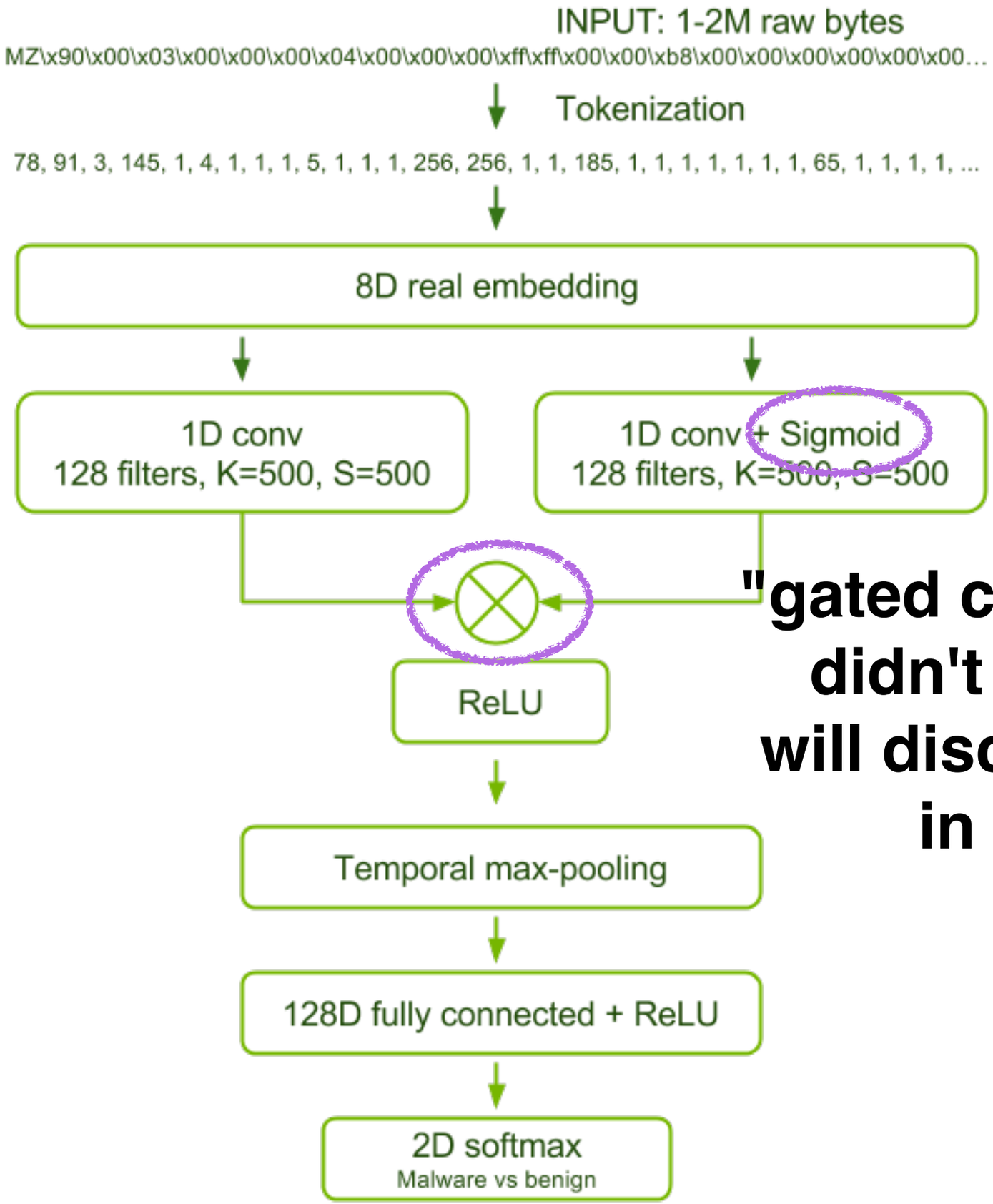
Posted on November 21, 2017 by Jon Barker | 0 Comments

Tagged Deep Learning, Malware Detection

https://devblogs.nvidia.com/parallelforall/malware-detection-neural-networks/

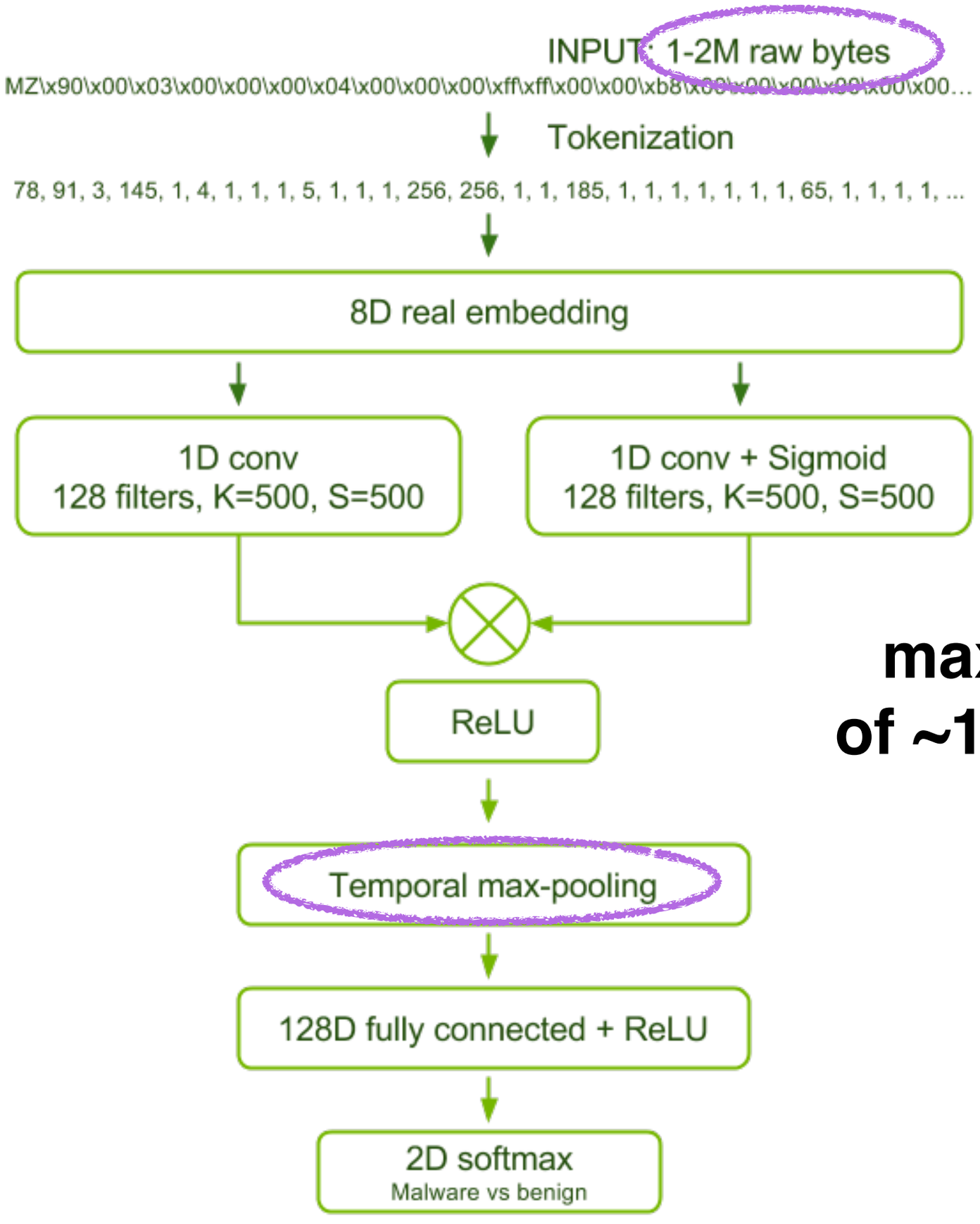


https://devblogs.nvidia.com/parallelforall/malware-detection-neural-networks/



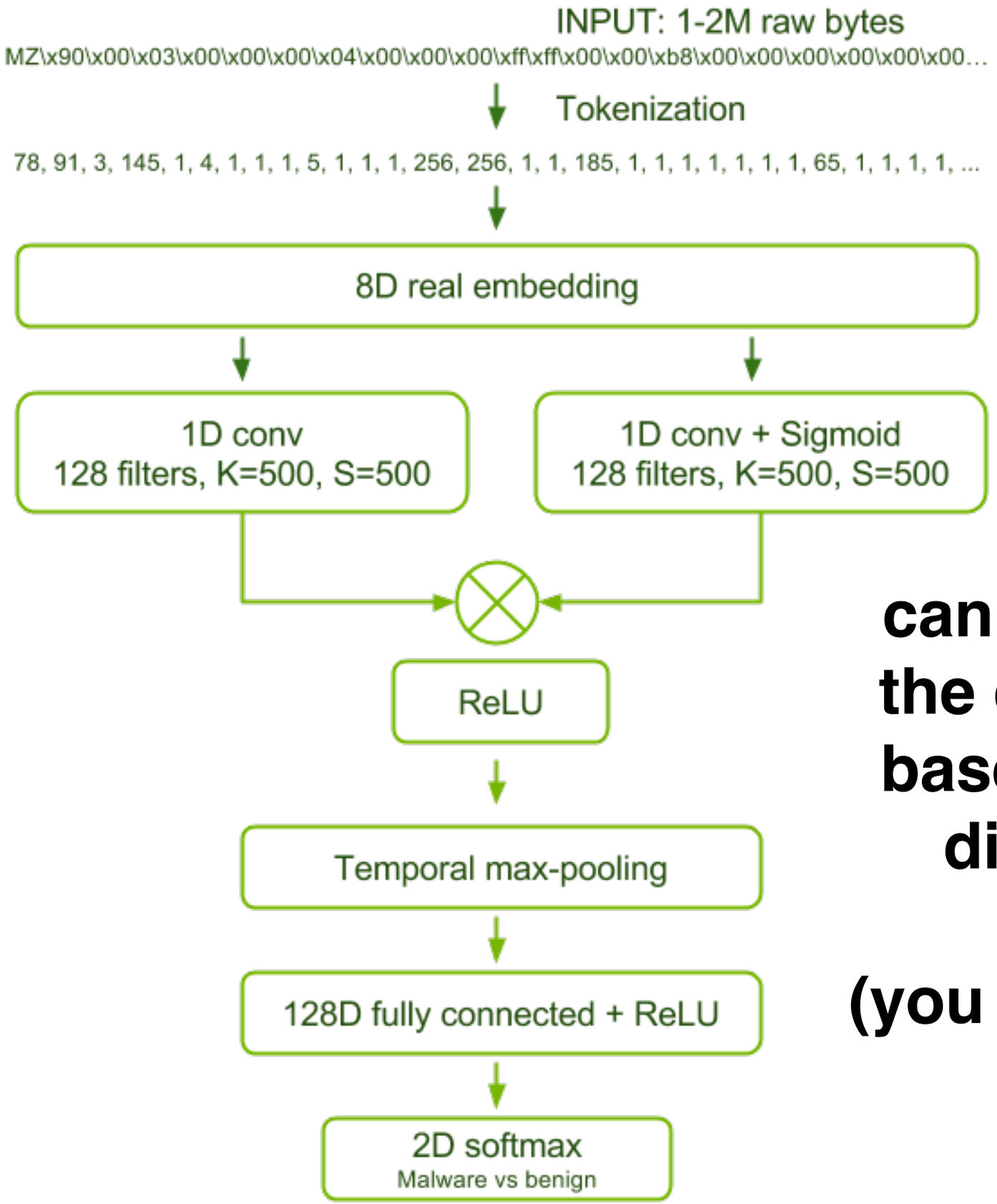
"gated convolution" didn't learn yet, will discuss gating in future.

https://devblogs.nvidia.com/parallelforall/malware-detection-neural-networks/



max pooling of ~1M vectors!

https://devblogs.nvidia.com/parallelforall/malware-detection-neural-networks/



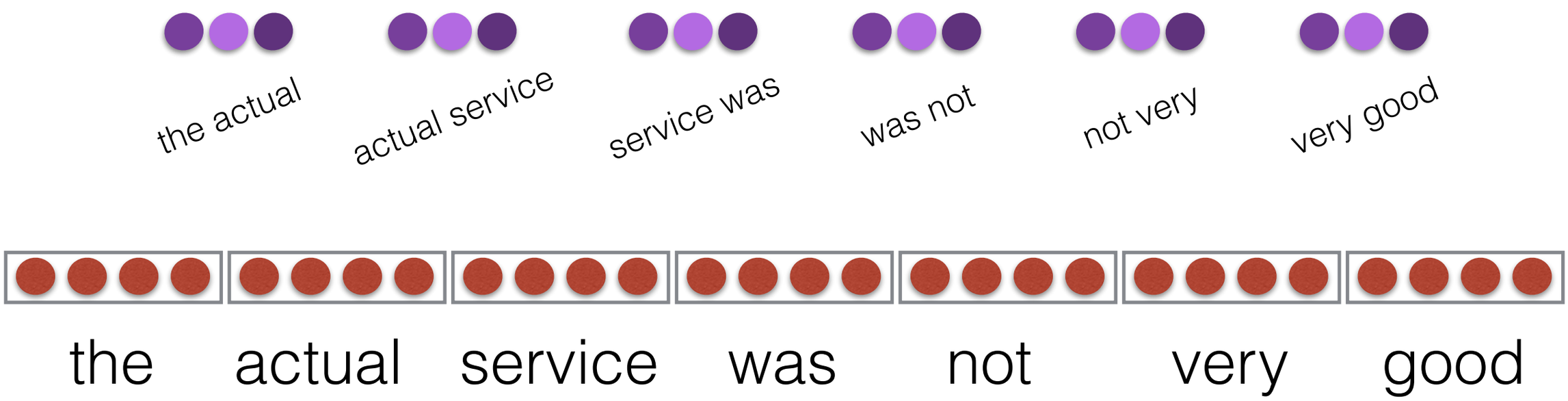
can you write the equations based on this diagram?

(you should be)

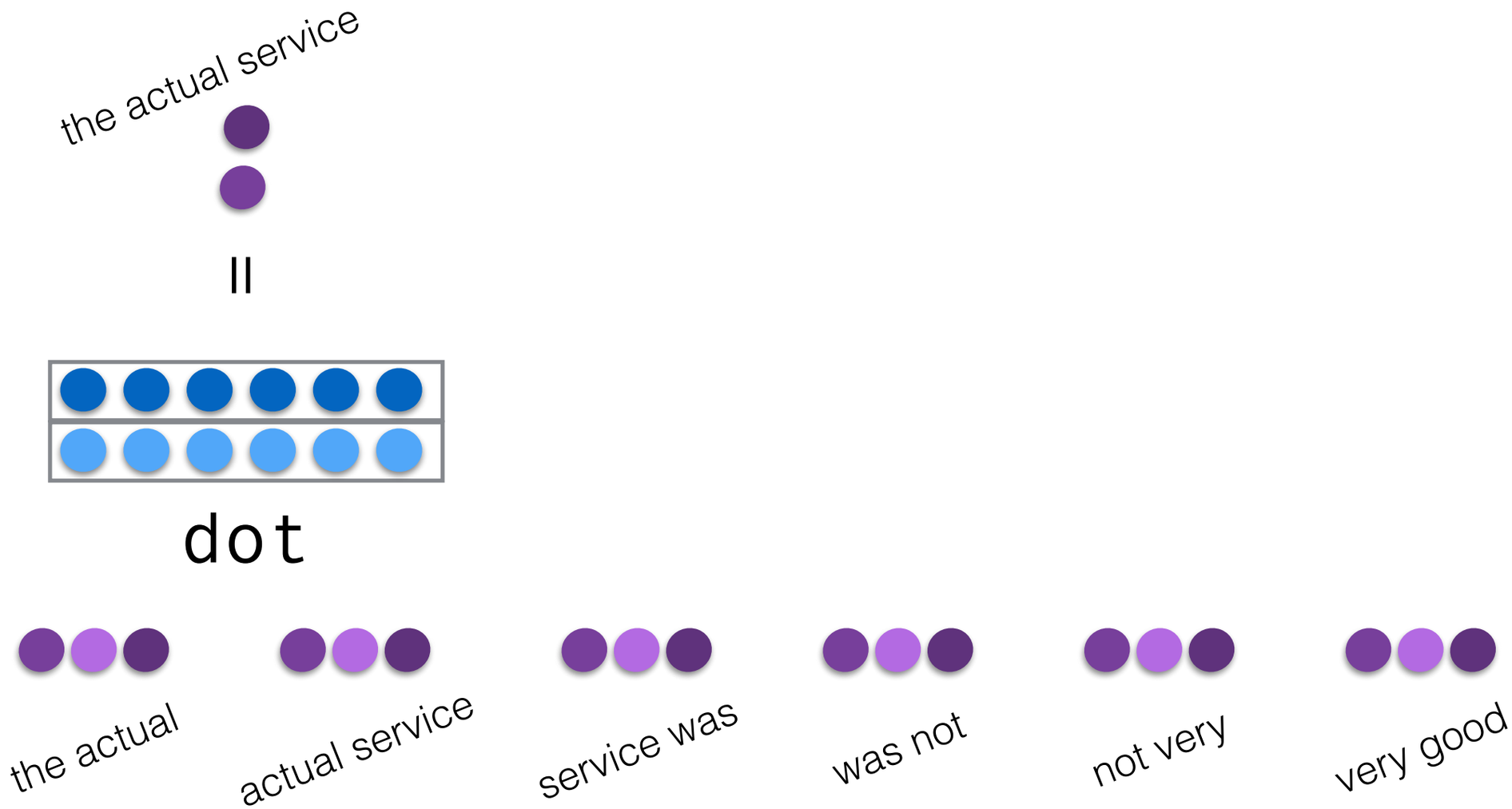
https://devblogs.nvidia.com/parallelforall/malware-detection-neural-networks/

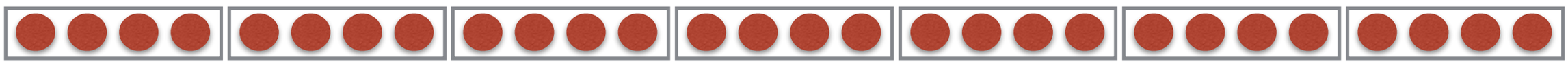
Hierarchy

Hierarchy

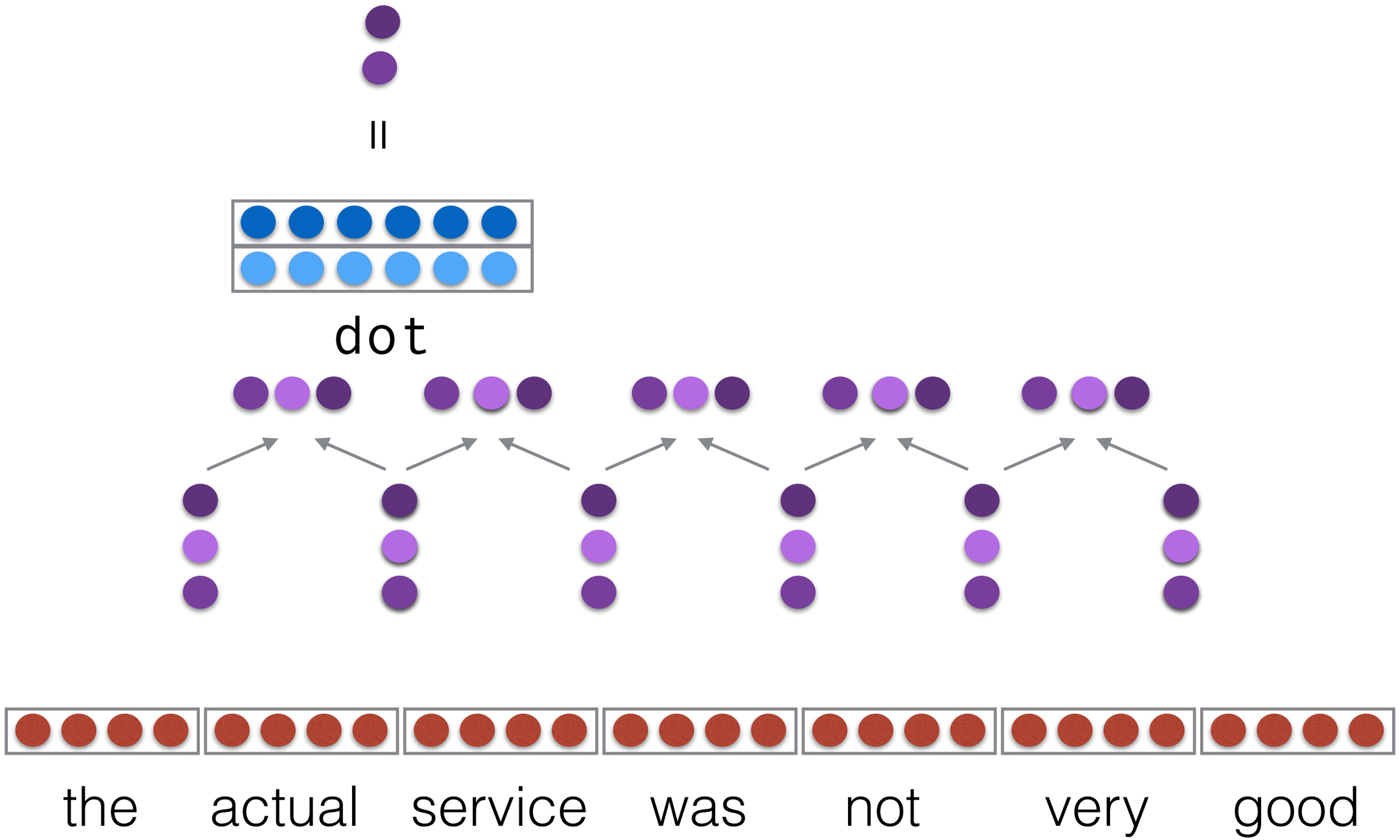


can have hierarchy



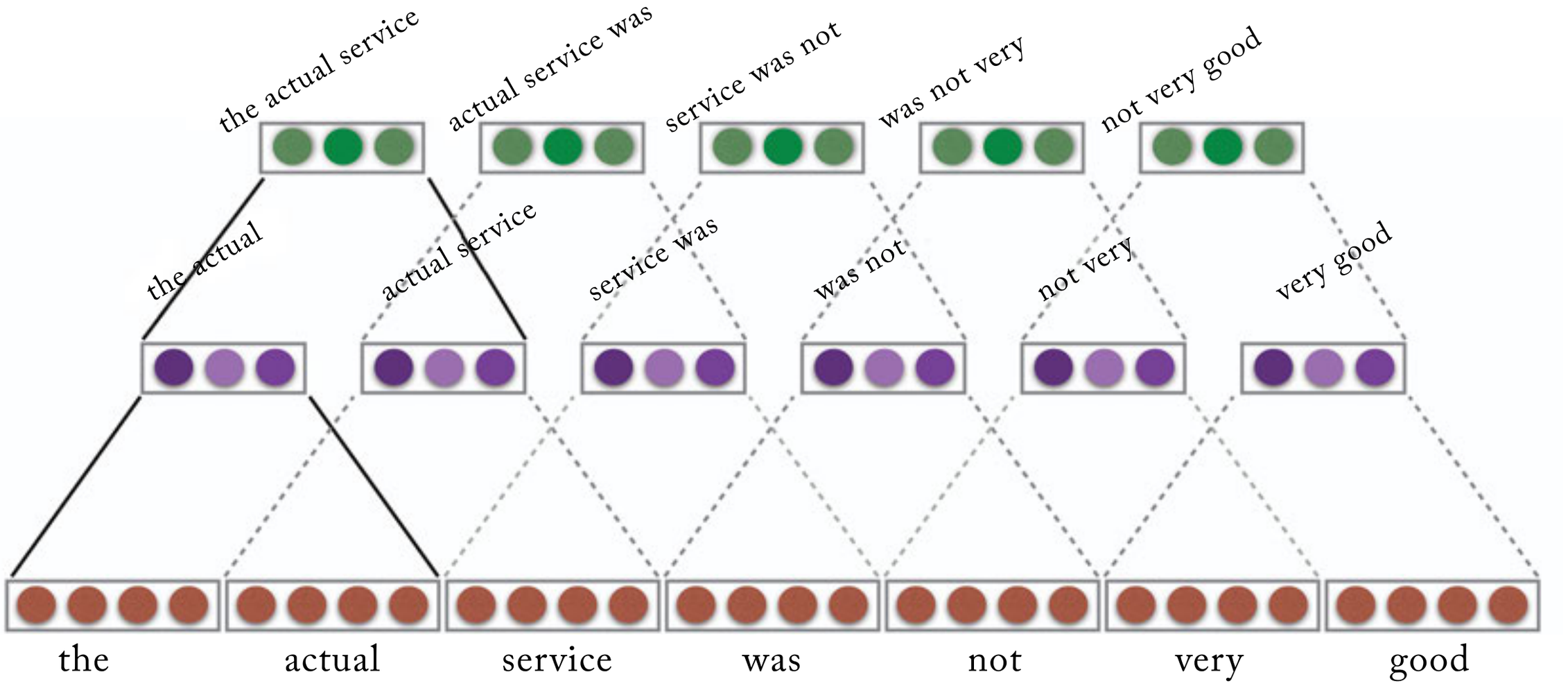


can have hierarchy



(can combine: **pooling + hierarchy)**

Hierarchy



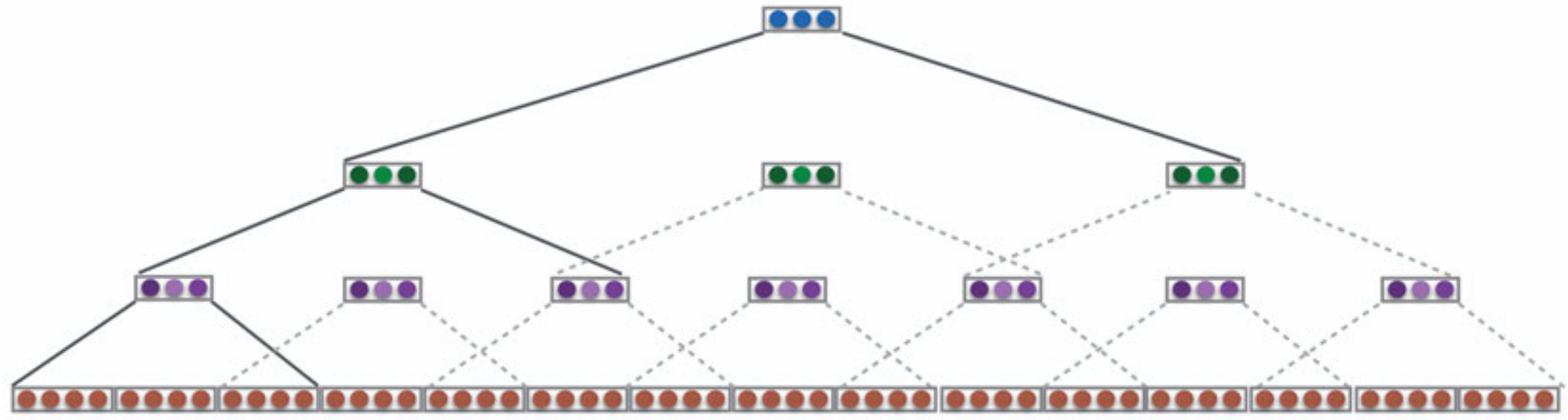
2-layer hierarchical conv with k=2

Dilated Convolutions

we want to cover more of the sequence

idea: strides + hierarchy

Dilated Convolutions



dilated convolution, k=3

idea: strides + hierarchy

ConvNets Summary

- Shared matrix used as feature detector.
- Extracts interesting ngrams.
- Pool ngrams to get fixed length representation.
- Max-pooling works well.
    - Max vs. Average pooling.
- Use hierarchy / dilation to expand coverage.
- Train end-to-end.

if time permits

Alternative: Hashing Trick

- ConvNet is an architecture for finding good ngrams.

- But if we know ngrams are important, why not just have ngram embeddings (ngram vectors)?

- --> for large vocabulary, not scalable.

  Can't represent all ngrams, don't know which are important.

Alternative: Hashing Trick

- **Problem**: our ngram vocabulary size if 10^9

- **Solution**: use smaller space via hashing,
  allow feature clashes.

Hashing Trick

- We have > 10^9 different ngrams.
- We can afford ~10^6 different embeddings.
- Map each ngram to a number in [0, 10^6]
- Use the corresponding embedding vector.
- Clashes will happen, but it will probably be ok.
    - Even safer: map each ngram to two numbers using two different hash functions, sum the vectors.

Hashing Trick vs ConvNets

- What are the benefits of using bag of ngrams?
- What are the benefits of using ConvNet (ngram detector)?
- Does it matter if the vocabulary size is small or large?

(discuss)