

Text classification

Based on lecture by Ekaterina Artemova



- **Problem statement**
- **Logistic regression model**
- **Simple document embeddings models**
- **Neural networks for classification, FastText**
- **Convolutional neural networks for text classification**
- **Data processing in text classification**



Problem statement



Problem statement

Data:

- $d \in D$ set of documents
- $c \in \mathcal{C}$ set of labels



Problem statement

Data:

- $d \in D$ set of documents
- $c \in C$ set of labels

Types of classification:

- binary classification $|C| = 2, \quad \forall d \in D \leftrightarrow c \in C$
- multiclass classification $|C| = K, K > 2, \quad \forall d \in D \leftrightarrow c \in C$
- multilabel classification $|C| = K, K > 2, \quad \forall d \in D \leftrightarrow C' \subseteq C$



Problem examples

- spam detection (binary)
 $C = \{\text{spam}, \text{not spam}\}$



Problem examples

- spam detection (binary)
 $C = \{\text{spam}, \text{not spam}\}$

Здравствуйте!

До 30 апреля по всей России
объявлены нерабочие дни.

Мы верим, что Вы очень ответственно
отнеслись к сложившейся ситуации.
Надеемся, что Вы планируете остаться
дома со своей семьей и
минимизировать внешние контакты.
Это важно именно сейчас, чтобы
защитить себя и других.

Меня зовут Бакаре Тунде, я брат
первого нигерийского космонавта,
майора ВВС Нигерии Абака Тунде. Мой
брат стал первым африканским
космонавтом, который отправился с
секретной миссией на советскую
станцию «Салют-6» в далеком 1979
году.



Problem examples

- Sentiment analysis (multiclass)
 $C = \{\text{negative}, \text{positive}, \text{neutral}\}$



Problem examples

- Sentiment analysis (multiclass)
 $C = \{\text{negative}, \text{positive}, \text{neutral}\}$

Пончик или пышка — круглое или кольцеобразное, жаренное во фритюре хлебобулочное изделие, с начинкой или без неё. Пышка в узком смысле слова — пончик без начинки с дыркой посередине.

Обожаю пончики на завтрак! Это лучшее начало трудового дня, они заряжают энергией и хорошим настроением меня и всю мою семью.
Гомер С.

Никогда больше не стану покупать у вас пончики. В пончике должна быть ОДНА дырка посередине, а в вашем пончике я нашла 15! Ну это никуда не годится!



Problem examples

- Articles tagging (multilabel)
 $C = \{\text{sport, politics, health, ...}\}$



Problem examples

- Articles tagging (multilabel)
 $C = \{\text{sport, politics, health, ...}\}$

«Зенит» на своем YouTube-канале провел онлайн-трансляцию тренировки футболистов в самоизоляции. Занятие прошло под руководством тренера по физподготовке Андреа Сканавино.

Посольство Северной Кореи в РФ поблагодарило КПРФ за телеграмму, направленную партией северокорейскому лидеру Ким Чен Ыну по случаю годовщины саммита РФ и КНДР, сообщил РИА Новости депутат от КПРФ Казбек Тайсаев.



Quality metrics for binary classification

Real classes

Predicted
classes

	Class 0	Class 1
Class 0	True positive (tp)	False positive (fp)
Class 1	False negative (fn)	True negative (tn)



Quality metrics for binary classification

Real classes

Predicted
classes

	Class 0	Class 1
Class 0	True positive (tp)	False positive (fp)
Class 1	False negative (fn)	True negative (tn)

- Accuracy

$$Acc = \frac{tp+tn}{tp+tn+fp+fn}$$



Quality metrics for binary classification

Real classes

Predicted
classes

	Class 0	Class 1
Class 0	True positive (tp)	False positive (fp)
Class 1	False negative (fn)	True negative (tn)

- Accuracy
- Precision

$$Acc = \frac{tp+tn}{tp+tn+fp+fn}$$

$$Pr = \frac{tp}{tp+fp}$$



Quality metrics for binary classification

Real classes

Predicted
classes

	Class 0	Class 1
Class 0	True positive (tp)	False positive (fp)
Class 1	False negative (fn)	True negative (tn)

- Accuracy
- Precision
- Recall

$$Acc = \frac{tp+tn}{tp+tn+fp+fn}$$

$$Pr = \frac{tp}{tp+fp}$$

$$R = \frac{tp}{tp+fn}$$



Quality metrics for binary classification

Real classes

Predicted
classes

	Class 0	Class 1
Class 0	True positive (tp)	False positive (fp)
Class 1	False negative (fn)	True negative (tn)

- Accuracy
- Precision
- Recall
- F-score

$$Acc = \frac{tp+tn}{tp+tn+fp+fn}$$

$$Pr = \frac{tp}{tp+fp}$$

$$R = \frac{tp}{tp+fn}$$

$$F_1 = \frac{2}{1/Pr+1/R} = \frac{2 \cdot Pr \cdot R}{Pr+R}$$



Quality metrics for binary classification

Real classes

Predicted
classes

	Class 0	Class 1	Class 2
Class 0	tn_{00}	fn_{10}	tn_{02}
Class 1	fp_{10}	tp_1	fp_{12}
Class 2	tn_{20}	fn_{12}	tn_{22}



Quality metrics for binary classification

Real classes

Predicted
classes

	Class 0	Class 1	Class 2
Class 0	tn_{00}	fn_{10}	tn_{02}
Class 1	fp_{10}	tp_1	fp_{12}
Class 2	tn_{20}	fn_{12}	tn_{22}

- Micro-averaging

$$Pr_{micro} = \frac{\sum tp_i}{\sum tp_i + \sum \sum fp_{ij}}$$

$$R_{micro} = \frac{\sum tp_i}{\sum tp_i + \sum \sum fn_{ij}}$$



Quality metrics for binary classification

Predicted
classes

Real classes

	Class 0	Class 1	Class 2
Class 0	tn_{00}	fn_{10}	tn_{02}
Class 1	fp_{10}	tp_1	fp_{12}
Class 2	tn_{20}	fn_{12}	tn_{22}

- Micro-averaging

$$Pr_{micro} = \frac{\sum tp_i}{\sum tp_i + \sum \sum fp_{ij}}$$

$$R_{micro} = \frac{\sum tp_i}{\sum tp_i + \sum \sum fn_{ij}}$$

- Macro-averaging

$$Pr_{macro} = \frac{\sum Pr_i}{|C|}$$

$$R_{macro} = \frac{\sum R_i}{|C|}$$



Features for text classification

- Features for classification can be extracted from text:
 - smileys count
 - words (and n-grams!)
 - syntactic structure



Features for text classification

- Features for classification can be extracted from text:
 - smileys count
 - words (and n-grams!)
 - syntactic structure
- External information can be useful, too:
 - author's age
 - number of citations



Text classification models

- Text classification can be solved by the same models, as the classical classification task:
 - linear models
 - metric models
 - decision trees
 - neural networks
 - ...



Text classification models

- Text classification can be solved by the same models, as the classical classification task:
 - linear models
 - metric models
 - decision trees
 - neural networks
- Model type can be chosen based on
 - data type and amount
 - performance requirements, complexity of customization and support



Multiclass classification

- Binary classification model can always be adapted to multiclass case
- Two possible strategies:



Multiclass classification

- Binary classification model can always be adapted to multiclass case
- Two possible strategies:
 - «One-vs-rest»
 - train a model for each class, the model can divide one class from the rest classes



Multiclass classification

- Binary classification model can always be adapted to multiclass case
- Two possible strategies:
 - «One-vs-rest»
 - train a model for each class, the model can divide one class from the rest classes
 - «One-vs-one»:
 - Train a classifier for each pair of classes
 - Useful for classifiers that work poorly with big data
 - Final class is chosen by majority vote



Main conclusions

- Text classification problem statement and approaches are similar to classical ones
- Main quality metrics are accuracy, precision, recall, F-score
- You can apply most of classical models
- Binary models can be adapted to multiclass case



Logistic regression model



Linear models

- Let x be a vector representation of a document, y - document class, w - a vector of linear model weights



Linear models

- Let x be a vector representation of a document, y - document class, w - a vector of linear model weights
- Model can be trained with different loss functions:
 - Logistic $\log(1 + \exp(-y\langle x, w \rangle))$
 - Hinge-loss $\max(0, 1 - y\langle x, w \rangle)$
 - Quadratic $\frac{1}{2} (y - \langle x, w \rangle)^2$



Linear models

- Let x be a vector representation of a document, y - document class, w - a vector of linear model weights
- Model can be trained with different loss functions
- Model can have different regularization:

- Lasso:

$$\alpha \|w\|_1$$

- Ridge:

$$\alpha \|w\|_2$$

- ElasticNet:

$$\alpha \|w\|_1 + \beta \|w\|_2$$



Logistic regression

- Linear model with logistic loss function - simple and good solution for text classification
- Model output:

$$p(c = 1|d) = \frac{1}{1+\exp(-z)}, \quad z = \omega^T x$$

w - a vector of linear model weights

x - a vector representation of a document

In multiclass case we replace logistic function with **softmax**



Logistic regression

Advantages

- Fast training
- Interpretability
- Easy inference

Disadvantages

- Linear model can't cope with difficult cases (and non-linear dependencies)



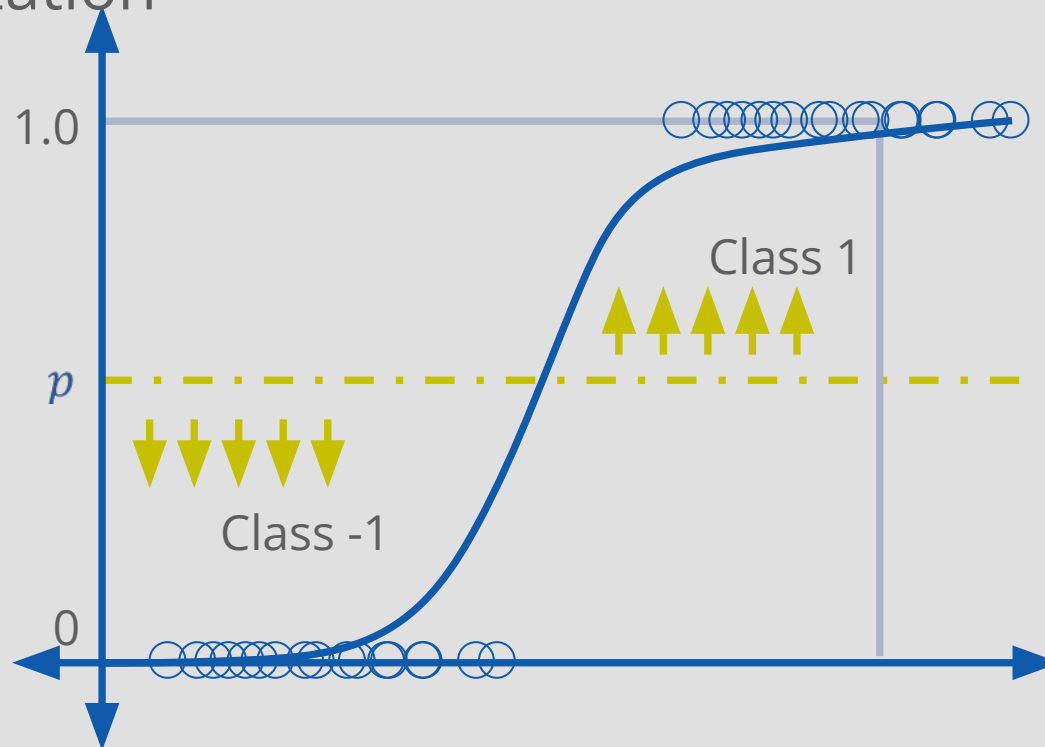
Threshold binarization

- Output of logistic regression is a probability of class 1
- You should choose a threshold for probability binarization



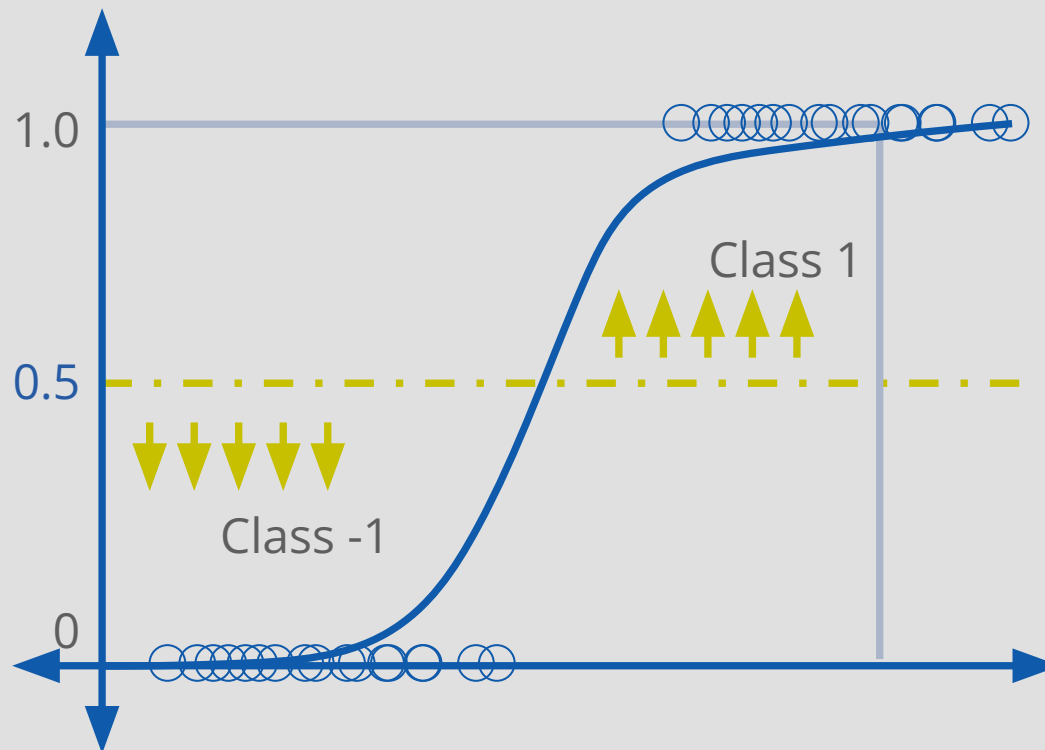
Threshold binarization

- Output of logistic regression is a probability of class 1
- You should choose a threshold for probability binarization



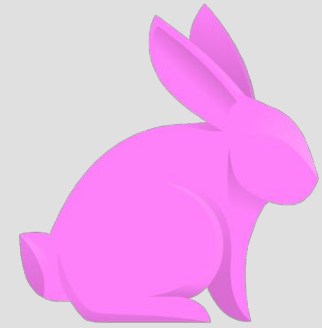
Threshold binarization

- Standard threshold is 0.5 but in order to take into account the specifics of the data it is better to select the threshold on the validation set



Vowpal Wabbit

- An efficient cross-platform tool for training linear models on big data
- Originally a product of Yahoo!, now Microsoft
- Supports basic loss functions, regularization

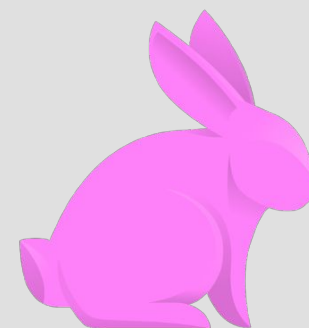


vowpalwabbit.org



Vowpal Wabbit

- An efficient cross-platform tool for training linear models on big data
- Originally a product of Yahoo!, now Microsoft
- Supports basic loss functions, regularization
- Uses several optimization methods, main one - SGD
- Trains model in streaming, constantly retrains model on new data
- Uses [hashing trick](#) for RAM usage optimization memory
- Can be applied as console application, or C++, C#, Java and Python package



vowpalwabbit.org



Main conclusions

- Logistic regression model is often used for text classification
- It is important to select the binarization threshold after model training
- Vowpal Wabbit allows to train linear models on large-scale data



Simple document embeddings models



Basic text representations

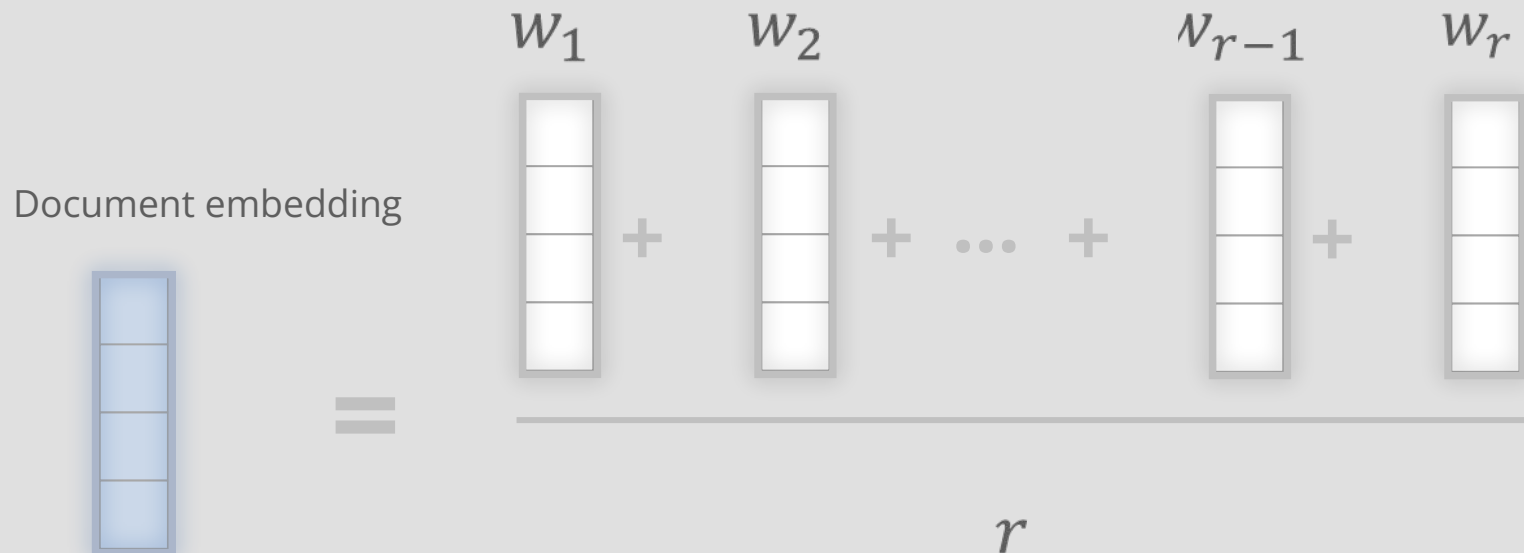
REMINDER

- Document from collection size m with a vocabulary with size n can be represented with an embedding of size n :
 - Bag of words
 - TF-IDF
 - SVD on a BoW matrix



Word embeddings averaging

- In order to obtain document embedding you can calculate the average of embeddings of words (such as [word2vec](#), [GloVe](#) or [FastText](#))



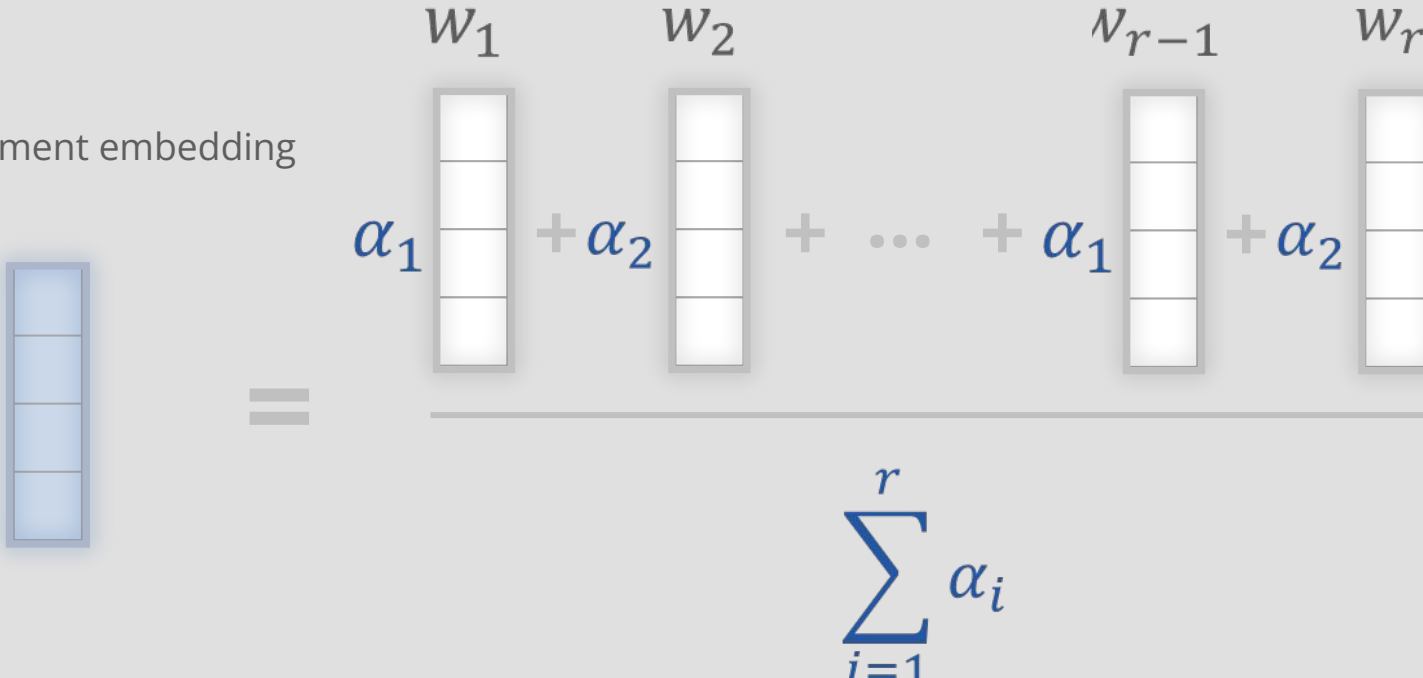
- It is simple and quite effective



Weighted average

- You can add weights into averaging

Document embedding



The diagram illustrates the calculation of a document embedding. On the left, a blue vertical rectangle represents the document embedding. To its right is an equals sign. Further right, a horizontal line serves as a base for a sum of weighted word embeddings. Above this line, four white vertical rectangles represent word embeddings, labeled w_1 , w_2 , w_{r-1} , and w_r from left to right. Below the first and third embeddings are the weights α_1 and α_1 respectively. Below the second and fourth embeddings are the weights α_2 and α_2 respectively. Plus signs and an ellipsis are placed between the embeddings to indicate a summation. Below the horizontal line, the summation formula $\sum_{i=1}^r \alpha_i$ is displayed.

$$\text{Document embedding} = \alpha_1 w_1 + \alpha_2 w_2 + \dots + \alpha_1 w_{r-1} + \alpha_2 w_r$$
$$\sum_{i=1}^r \alpha_i$$

- Weights can be obtained from TF-IDF matrix



Weighted average

- You can add weights into averaging
 - Weights can be obtained from TF-IDF matrix

Document embedding

The diagram illustrates the calculation of a document embedding. On the left, a blue vertical vector represents the 'Document embedding'. This is shown to be equal to a weighted sum of word embeddings. Two white vertical vectors, labeled w_1 and w_2 , represent individual word embeddings. They are multiplied by weights α_1 and α_2 respectively, and then summed together with an ellipsis indicating further terms. A horizontal line separates this sum from the summation formula below.

$$\sum_{i=1}^r \alpha_i$$

- Zero weights of stop-words and unimportant words allow us to obtain a good embedding even for long documents



Doc2vec model

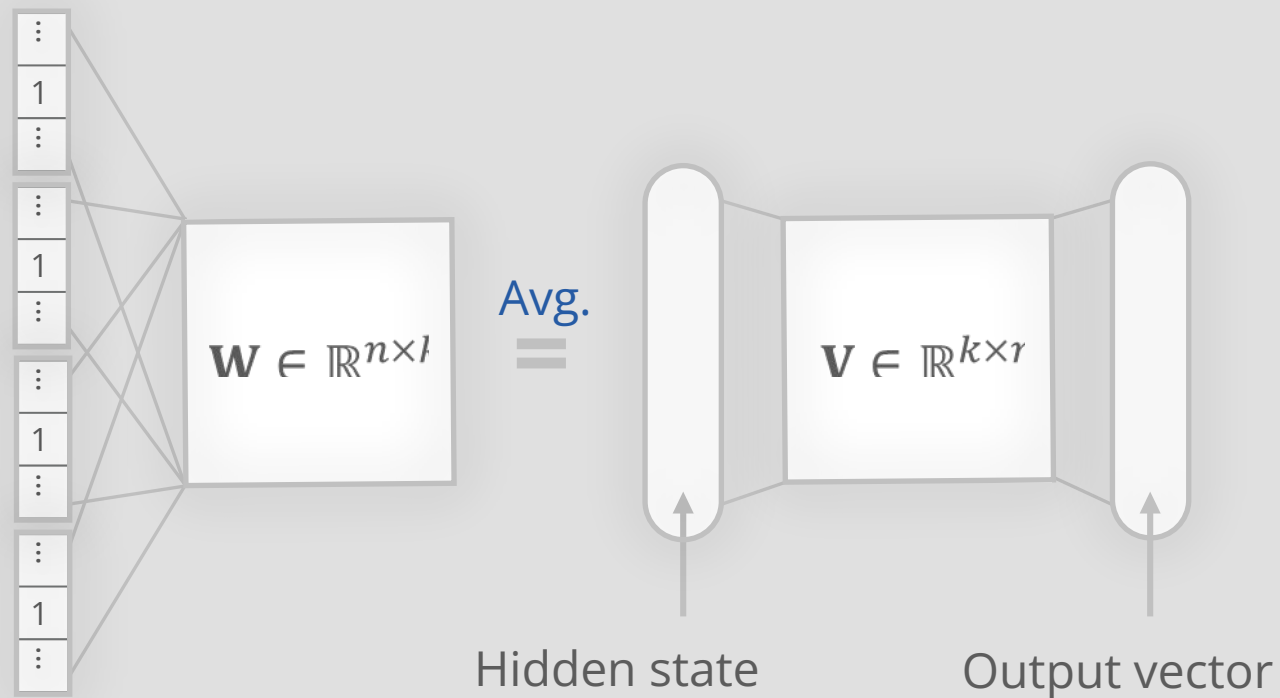
- Methods of training document embeddings similar to word2vec
- Two models: Distributed Memory and Distributed Bag of Words

- Quoc Le, Tomas Mikolov. *Distributed Representations of Sentences and Documents* // ICML. — 2014.



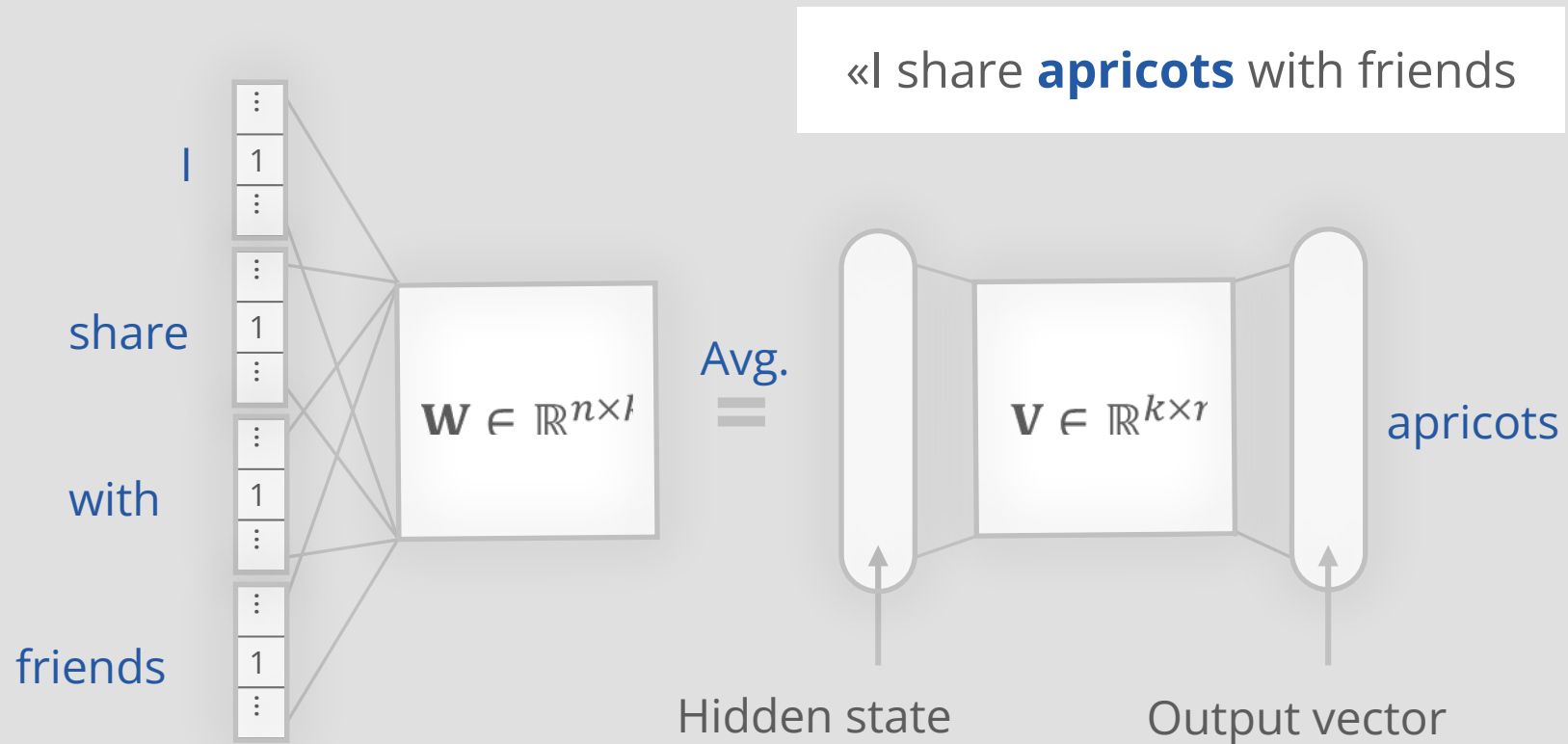
Distributed Memory doc2vec

- Similar to CBOW word2vec



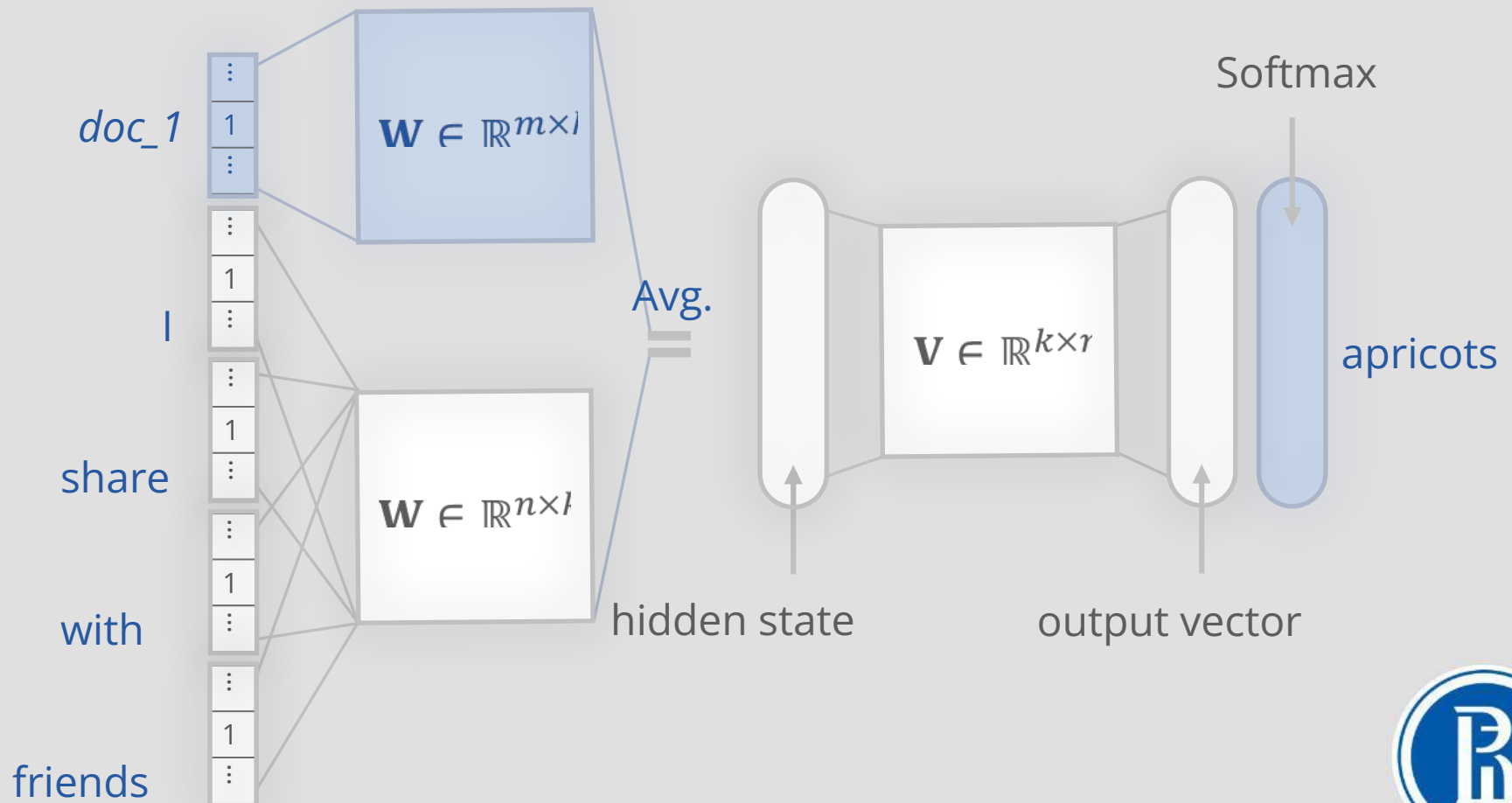
Distributed Memory doc2vec

- Similar to CBOW word2vec
- Slide with window over document words



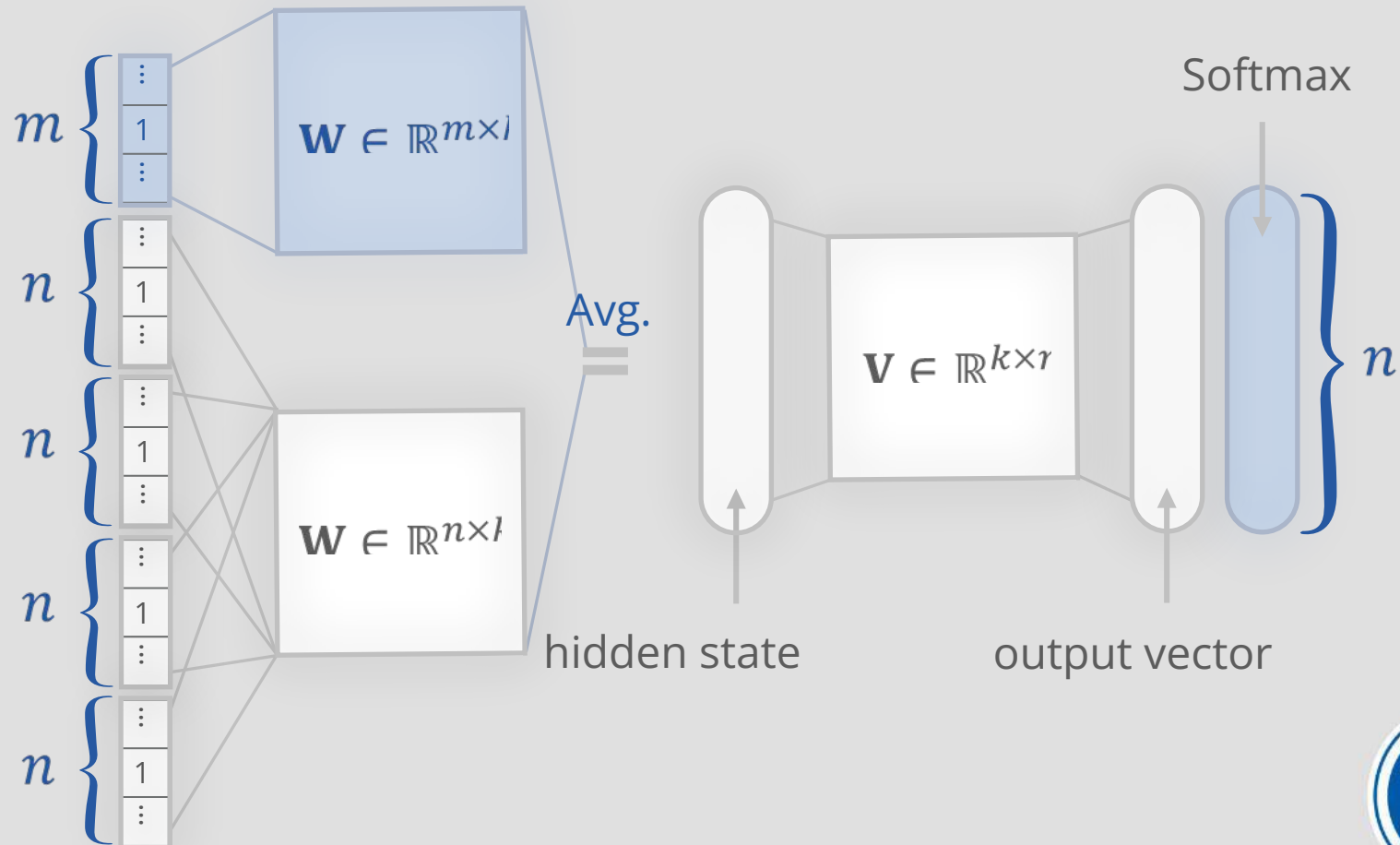
Distributed Memory doc2vec

- Along with words we use documents (paragraphs) in training
- Add one-hot vector sized m of document in each window



Distributed Memory doc2vec

- Along with words we use documents (paragraphs) in training
- Add one-hot vector sized m of document in each window



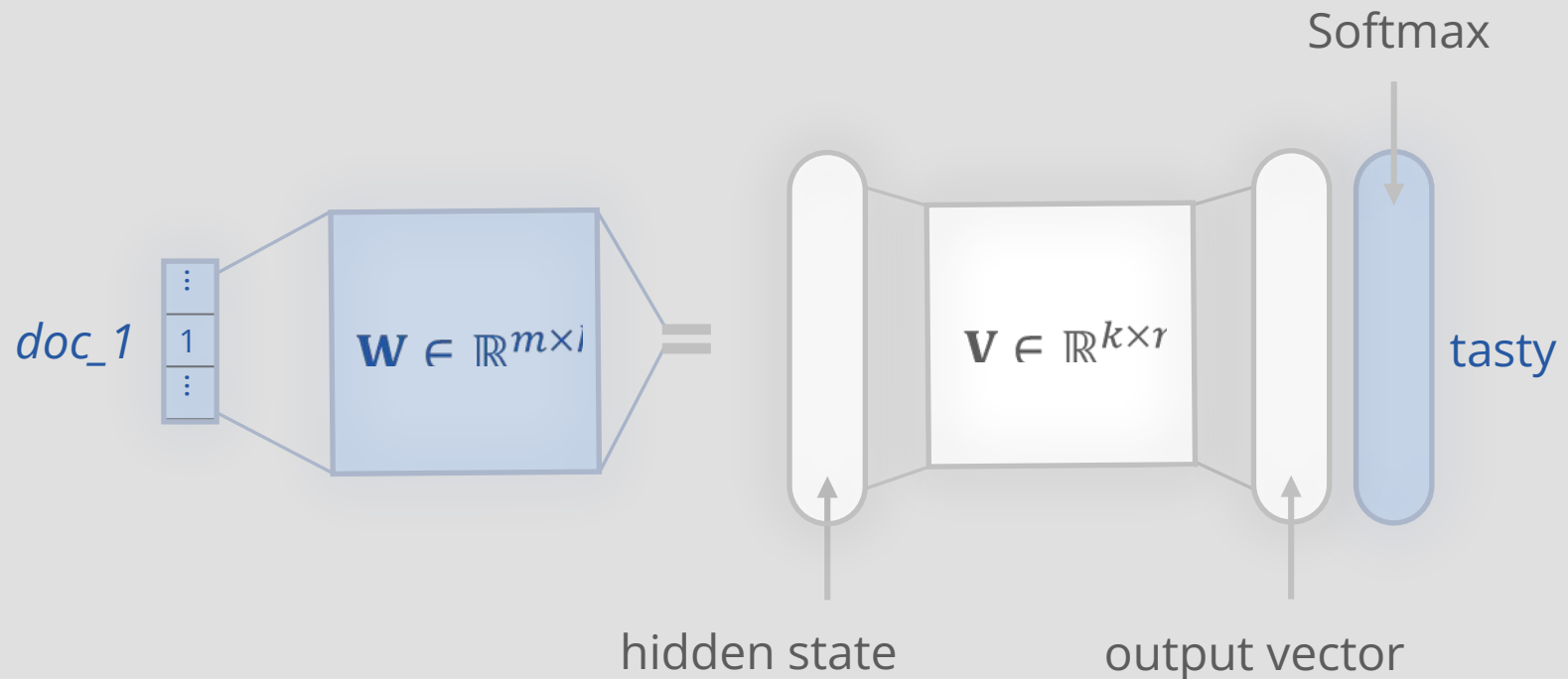
Distributed Bag of Words doc2vec

- Similar to Skip-gram word2vec
- Model is trained to predict random word from a random window based on one-hot embedding of a document, where the word comes from



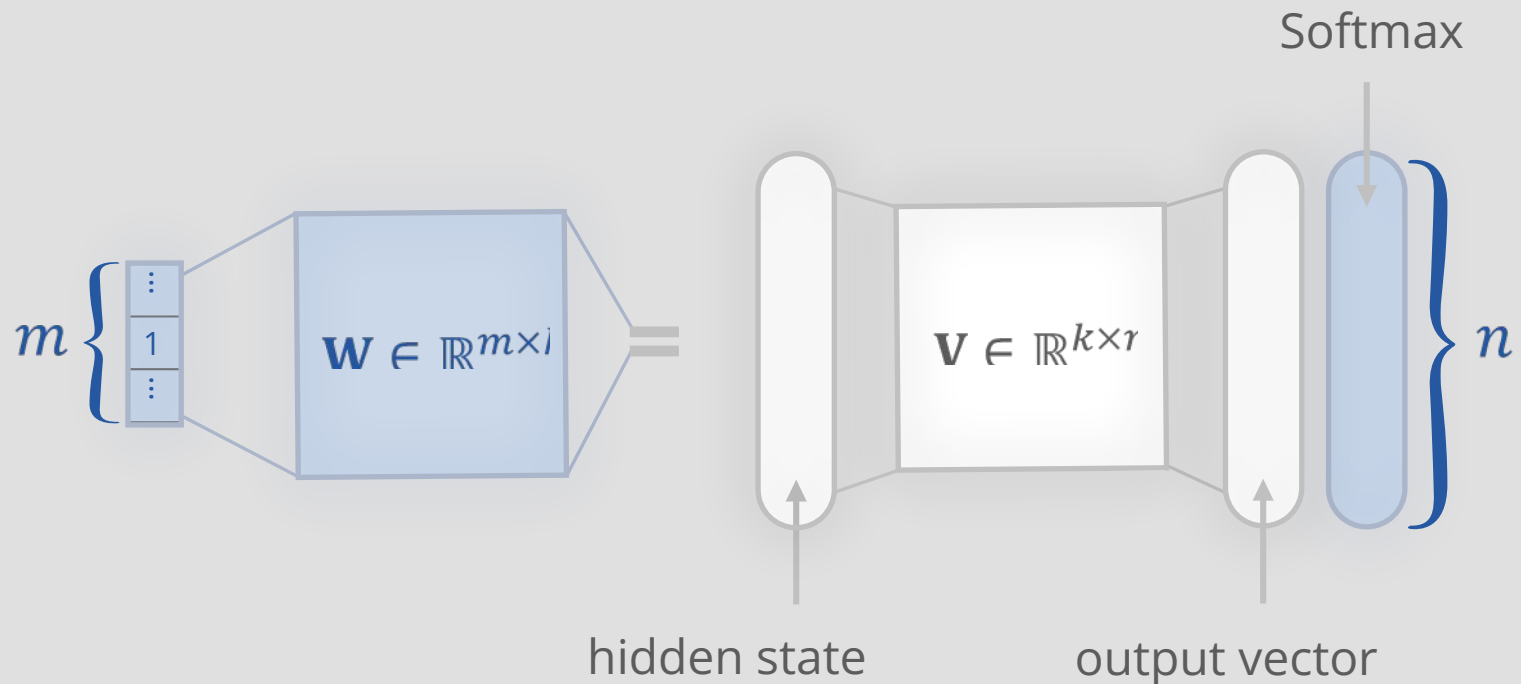
Distributed Bag of Words doc2vec

- Similar to Skip-gram word2vec
- Model is trained to predict random word from a random window based on one-hot embedding of a document, where there word comes from



Distributed Bag of Words doc2vec

- Similar to Skip-gram word2vec
- Model is trained to predict random word from a random window based on one-hot embedding of a document, where there word comes from



Main conclusions

- Basic document representations - Bow and TF-IDF
- SVD can be used for dimension reduction
- Averaging word embeddings is a common approach
- doc2vec model allows to obtain document embeddings



Neural networks for classification, FastText



Fully connected networks for classification

- Linear models can't cope with difficult dependencies
- Fully connected networks can be more difficult and flexible
- Due to large amount of weights such networks usually consist of 2-3 layers
- Input embedding can be pretrained or trained during the network training



Deep Averaging Network

- Suppose we have pretrained word embeddings
- We train a two-layer fully connected neural network to solve classification problem
- We calculate average of input embeddings as input
- To predict classes we calculate softmax

- *Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, Hal Daumé III. Deep Unordered Composition Rivals Syntactic Methods for Text Classification // IJCNLP. — 2015.*



Deep Averaging Network

- f - activation function
- W_i - weight matrix of layer i

only w_1

we w_2

have w_3

...

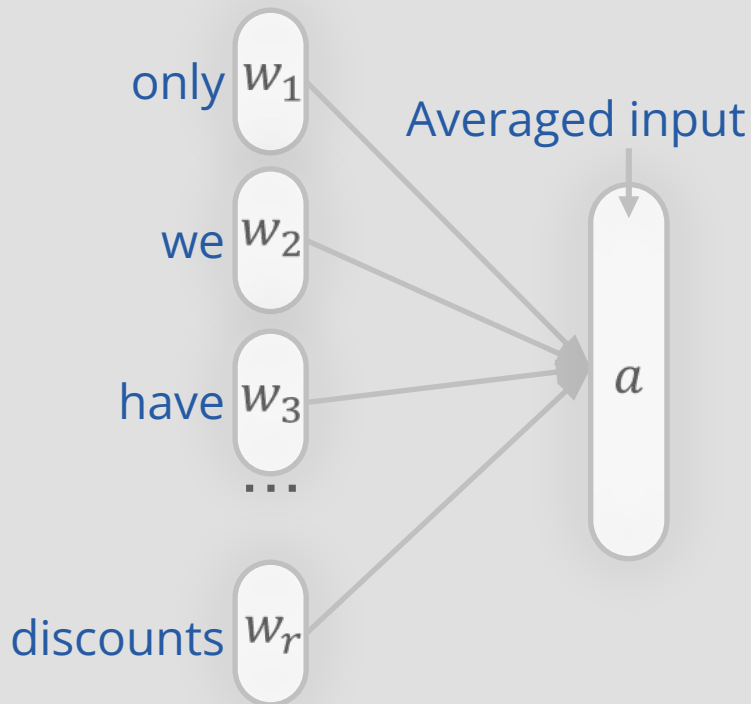
discounts w_r



Deep Averaging Network

$$a = \frac{1}{r} \sum_{i=1}^r w_i$$

- f - activation function
- W_i - weight matrix of layer i

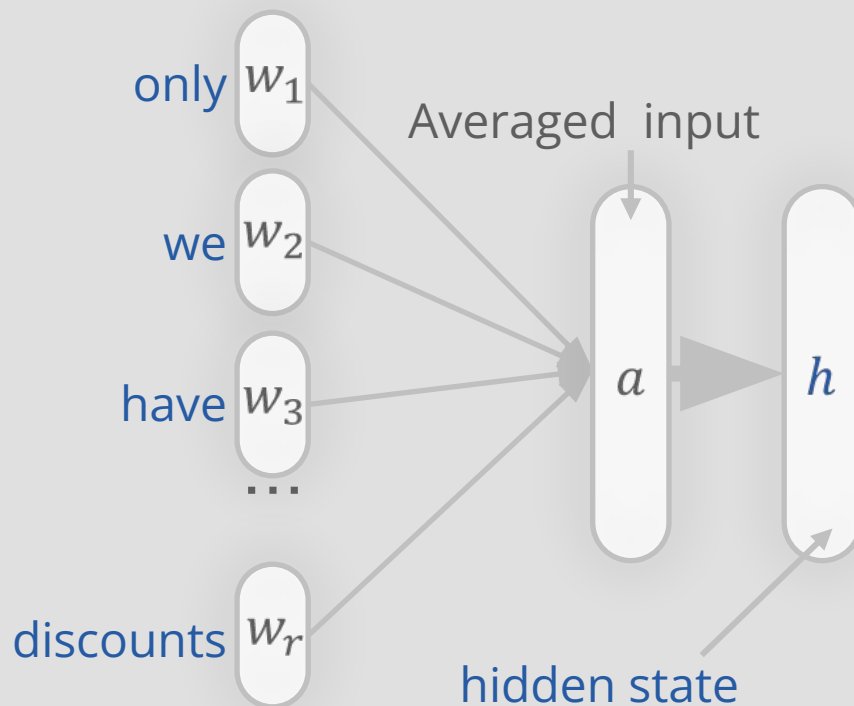


Deep Averaging Network

- f - activation function
- W_i - weight matrix of layer i

$$a = \frac{1}{r} \sum_{i=1}^r w_i$$

$$h = f(W_1^T a)$$



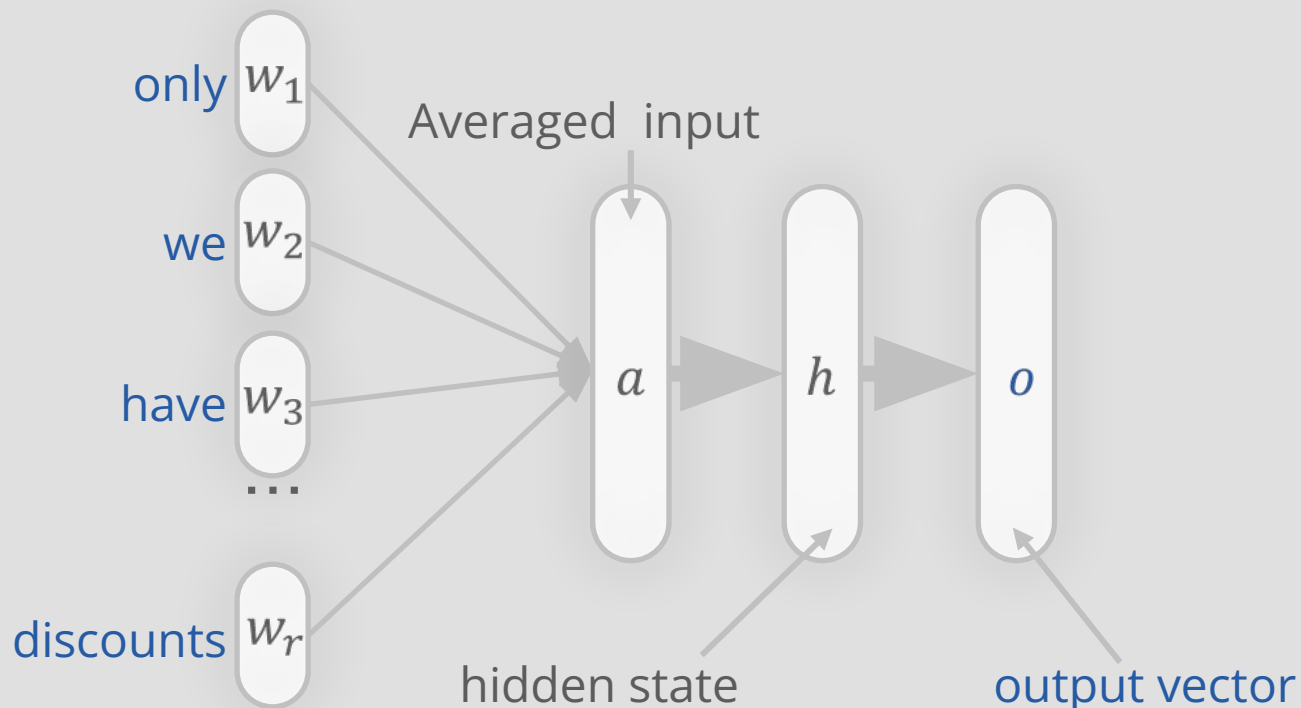
Deep Averaging Network

- f - activation function
- W_i - weight matrix of layer i

$$a = \frac{1}{r} \sum_{i=1}^r w_i$$

$$h = f(W_1^T a)$$

$$o = f(W_2^T h)$$



Deep Averaging Network

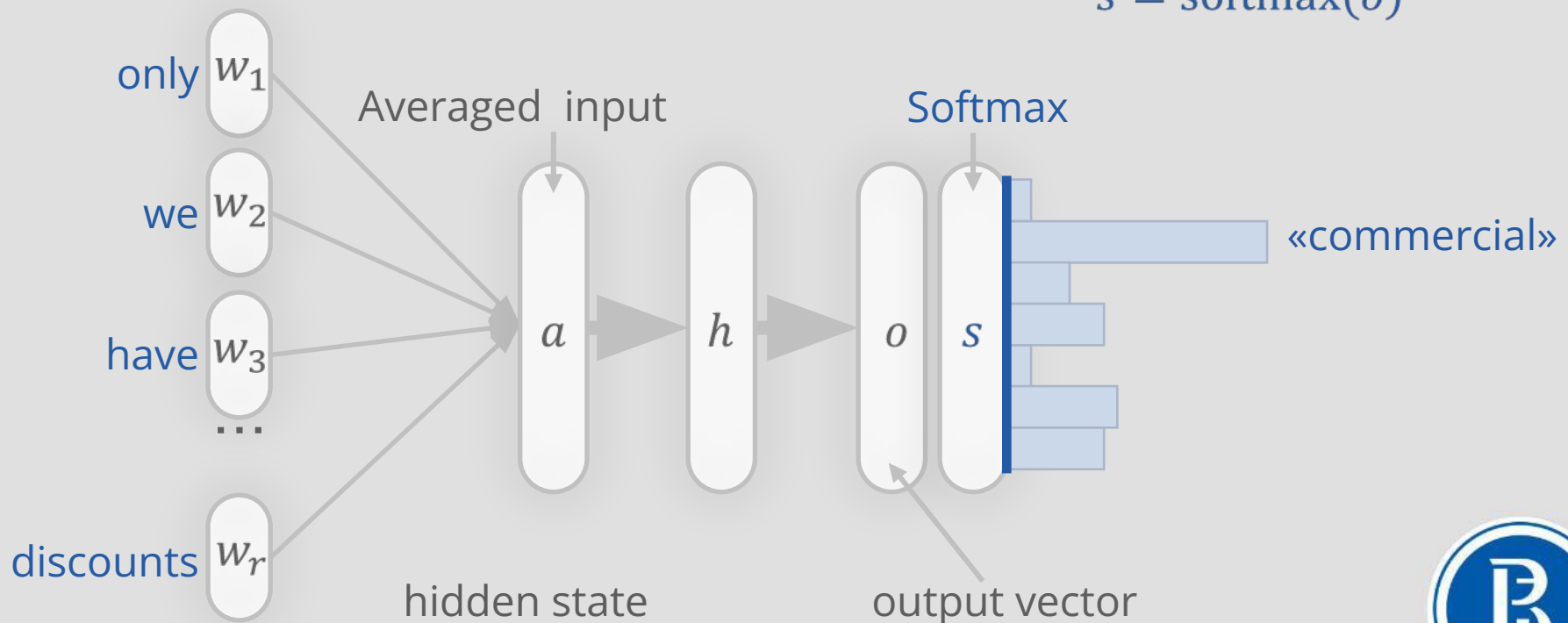
- f - activation function
- W_i - weight matrix of layer i

$$a = \frac{1}{r} \sum_{i=1}^r w_i$$

$$h = f(W_1^T a)$$

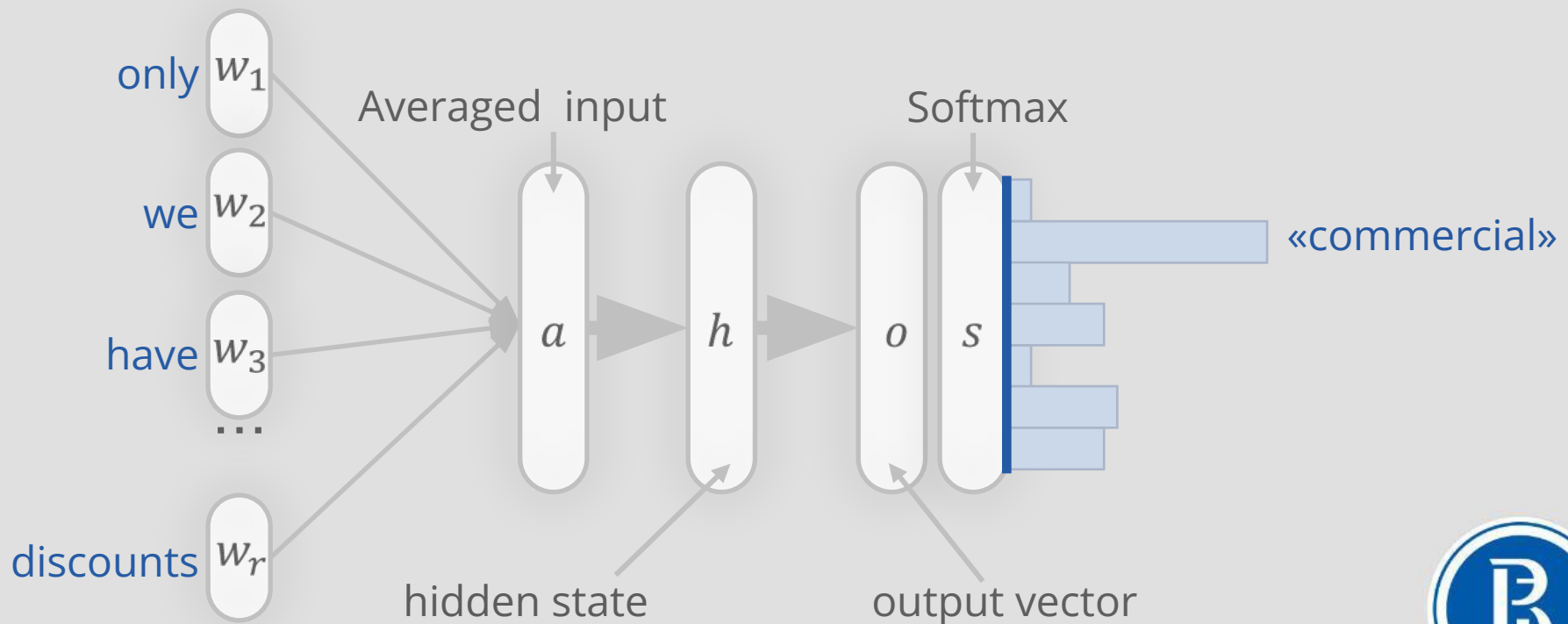
$$o = f(W_2^T h)$$

$$s = \text{softmax}(o)$$



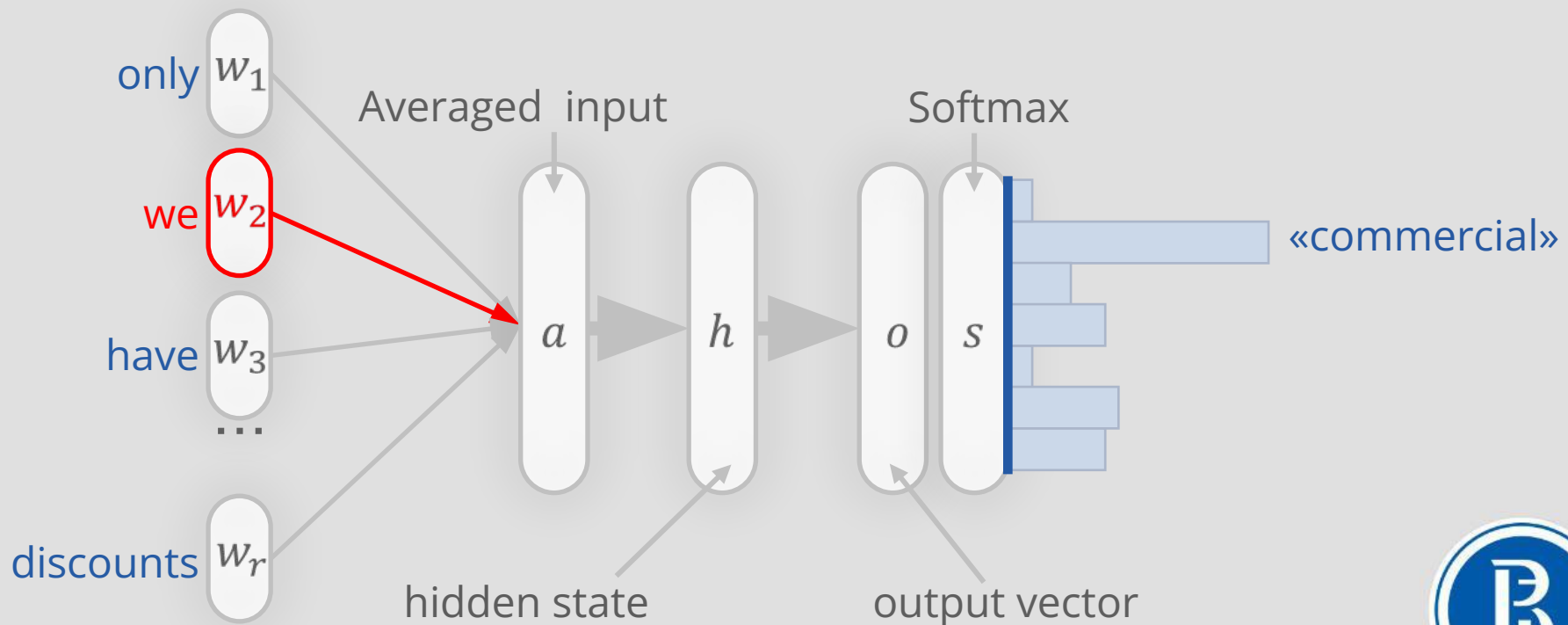
Word dropout

- Dropout regularization prevents from overfitting



Word dropout

- Dropout regularization prevents from overfitting
- Instead of vanishing outputs we will randomly exclude words from input



Deep Averaging Network

Advantages

- Is faster than complex architectures
- Often shows quality comparable to complex models
- Interpretable output probabilities

Disadvantages

- Does not take into account syntactic links, which can be partially corrected by using N-grams
- A large number of parameters compared to a linear classifier requires more training data



- FastText — already familiar library for getting vector representations
- Can work with **character n-grams**
- Optimizes RAM consumption with **hashing trick**
- Can be trained **in parallel** on CPU
- Has a console application version as well as **C++** and **Python package**



FastText as a classifier

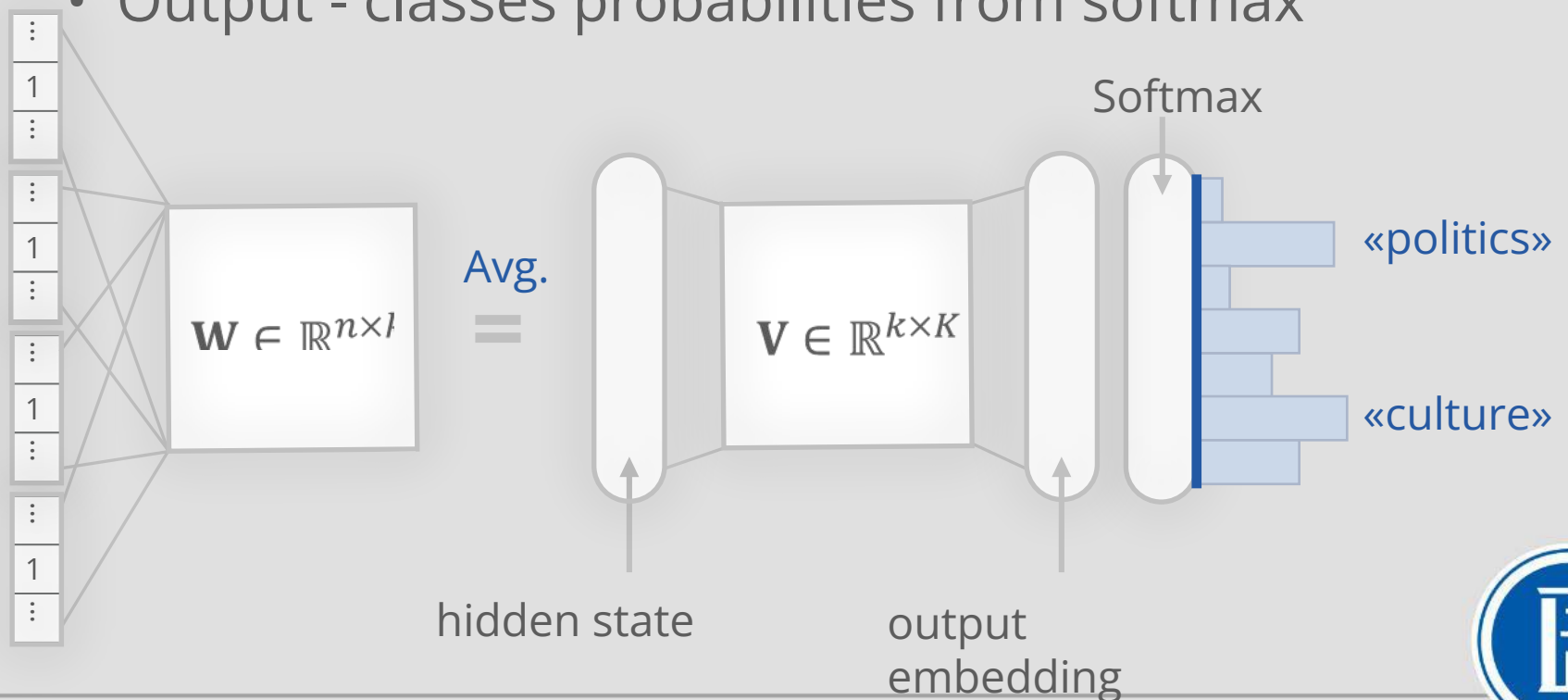
- The architecture remains the same
- The first layer is a matrix of weights that correspond to features: words, character and word n-grams
- The second layer is a K-classes classifier
- Both matrices are trained simultaneously
- Input - one-hot encoded vectors of features
- Output - classes probabilities from softmax

- *Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov. Bag of Tricks for Efficient Text Classification // EACL. — 2017.*



FastText as a classifier

- The first layer is a matrix of weights that correspond to features: words, character and word n-grams
- The second layer is a K-classes classifier
- Output - classes probabilities from softmax



Input data format

- Collection is a text file, each **line** — one **document**
- Each line contains **class label** and **text** (**no more than 1024 tokens**)

```
__label__review __label__negative i don't like your  
service literally can't stand it any more
```



Формат входных данных

- Collection is a text file, each **line** — one **document**
- Each line contains **class label** and **text** (**no more than 1024 tokens**)

```
__label__review __label__negative i don't like your  
service literally can't stand it any more
```

- **__label__** — a key word for class label
- text contains only raw words
- **N-grams** are collected automatically during training process



Main conclusions

- Texts can be classified with shallow neural networks
- DAN architecture includes several fully connected layers and an average embedding of words as an input
- Syntax can be partially taken into account through the use of dictionary N-gram features
- FastText can be trained quite fast on large-scale data without GPU
- FastText operates both with words and character and word n-grams



Convolutional neural networks for text classification



Convolutional Neural Network

- Convolutional neural networks (CNN) were a breakthrough in image processing
- It turns out that they work pretty well with texts, too
- In classification problem CNN allows to work with local word context without using word n-grams
- CNN can be trained much faster than recurrent networks and show comparable results in text classification



CNN components

- **Input layer** — contains features embeddings (word embeddings, for example)
- **Convolutional layer** — extracts local features from input embeddings
- **Discretization layer (pooling)** — extracts global features with maximal signal
- **Fully connected layer** — a classifier trained on resulting representations



Convolutional layer

- **Convolutional layer** — extracts local features from input embeddings
- Is based on convolution with **kernel**:

-1	0	1
-2	0	2
-1	0	1

- Input data is a **matrix**:

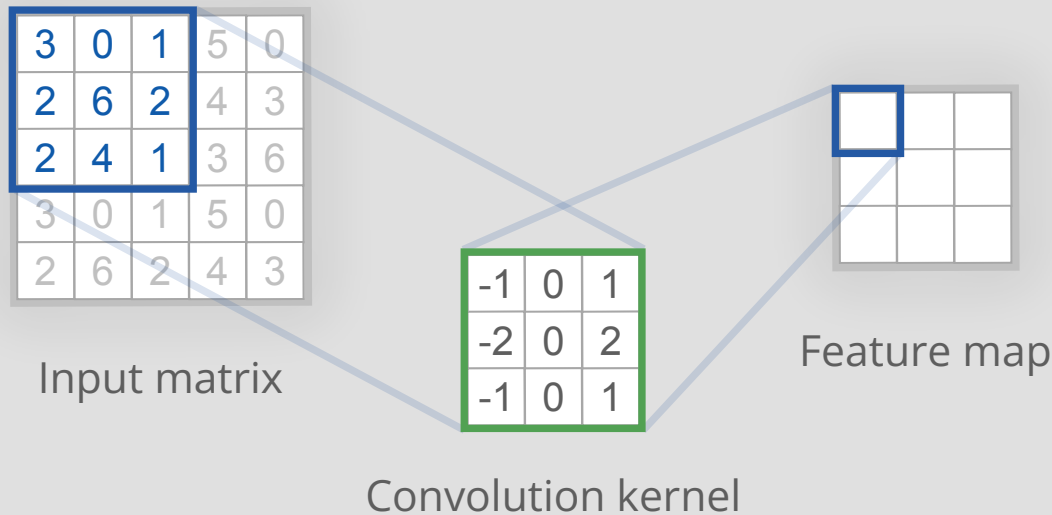
3	0	1	5	0
2	6	2	4	3
2	4	1	3	6
3	0	1	5	0
2	6	2	4	3

- Convolution is applied to fragments of the input matrix sequentially



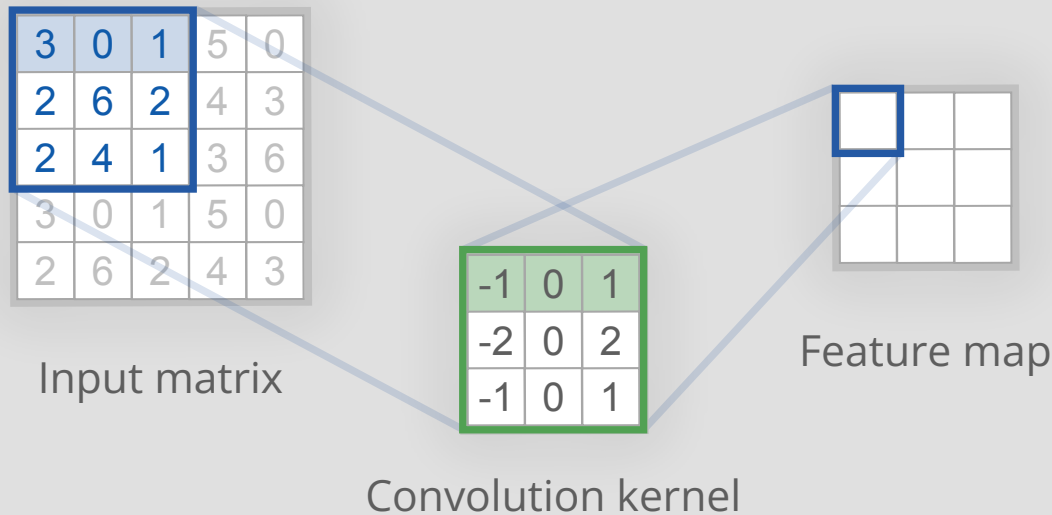
Convolution

- Convolution is applied to fragments of the input matrix sequentially:



Convolution

- Convolution is applied to fragments of the input matrix sequentially:

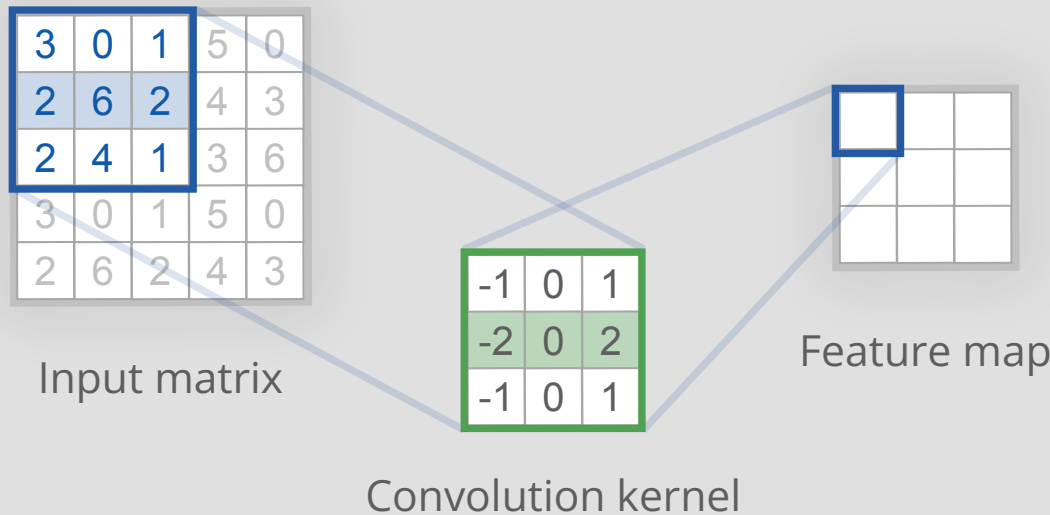


$$(-1 \times 3) + (0 \times 0) + (1 \times 1) +$$



Convolution

- Convolution is applied to fragments of the input matrix sequentially:

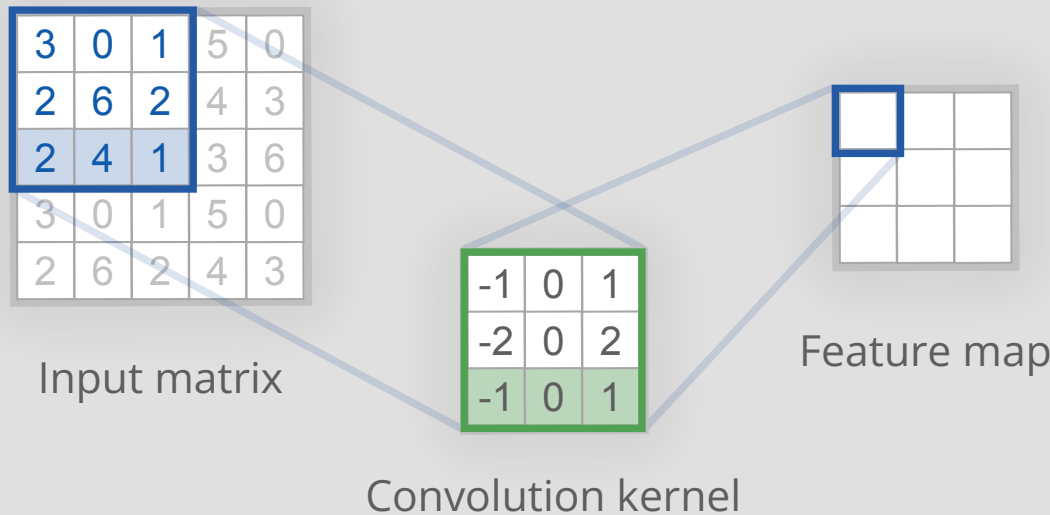


$$\begin{aligned} &(-1 \times 3) + (0 \times 0) + (1 \times 1) + \\ &(-2 \times 2) + (0 \times 6) + (2 \times 2) + \end{aligned}$$



Convolution

- Convolution is applied to fragments of the input matrix sequentially:

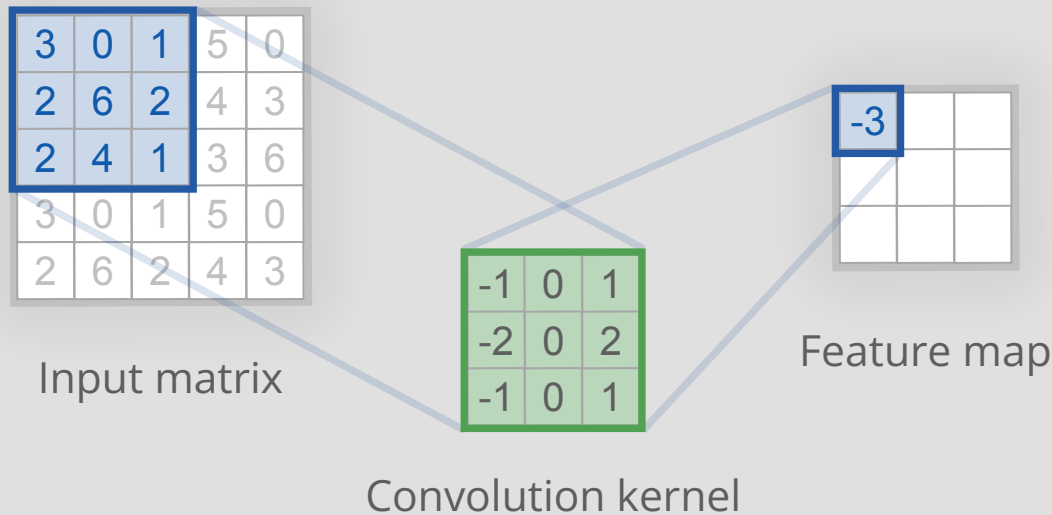


$$\begin{aligned} &(-1 \times 3) + (0 \times 0) + (1 \times 1) + \\ &(-2 \times 2) + (0 \times 6) + (2 \times 2) + \\ &(-1 \times 2) + (0 \times 4) + (1 \times 1) \end{aligned}$$



Convolution

- Convolution is applied to fragments of the input matrix sequentially:

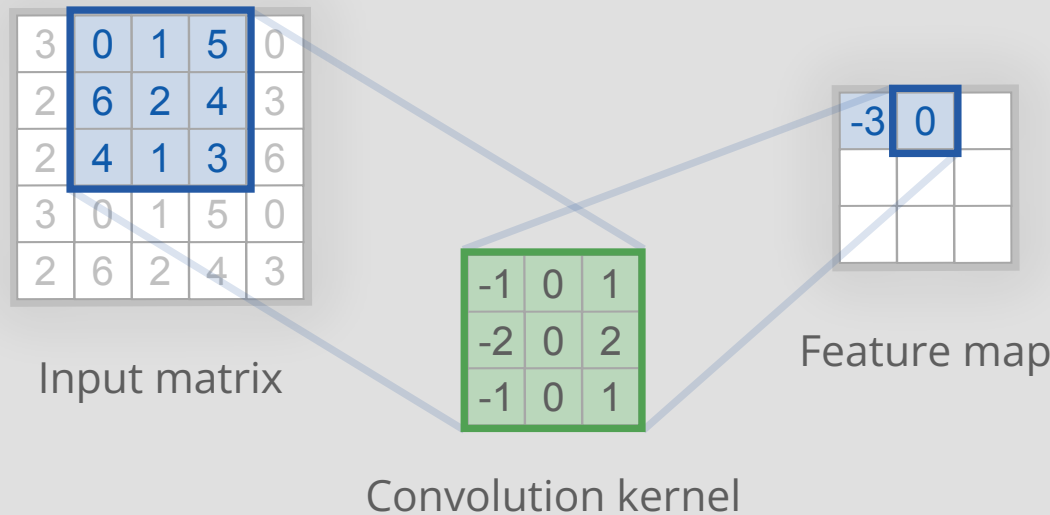


$$\begin{aligned} &(-1 \times 3) + (0 \times 0) + (1 \times 1) + \\ &(-2 \times 2) + (0 \times 6) + (2 \times 2) + \\ &(-1 \times 2) + (0 \times 4) + (1 \times 1) = -3 \end{aligned}$$



Convolution

- Convolution is applied to fragments of the input matrix sequentially:

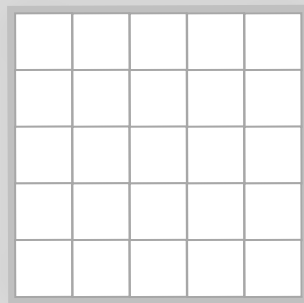


$$\begin{aligned} &(-1 \times 0) + (0 \times 1) + (1 \times 5) + \\ &(-2 \times 6) + (0 \times 2) + (2 \times 4) + \\ &(-1 \times 4) + (0 \times 1) + (1 \times 3) = 0 \end{aligned}$$

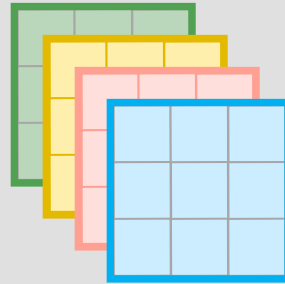


Convolutional layer

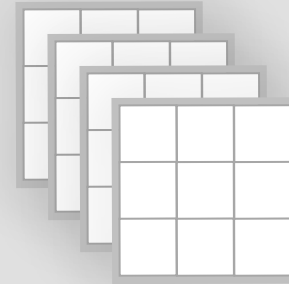
- Several kernels (filters) are applied in each layer
- Each filter creates its own feature map:



Input matrix



Convolution kernels

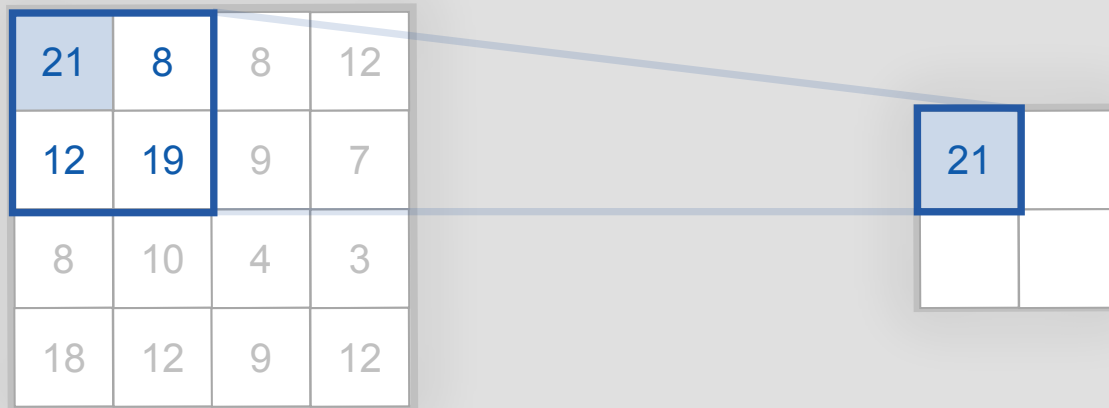


Feature maps



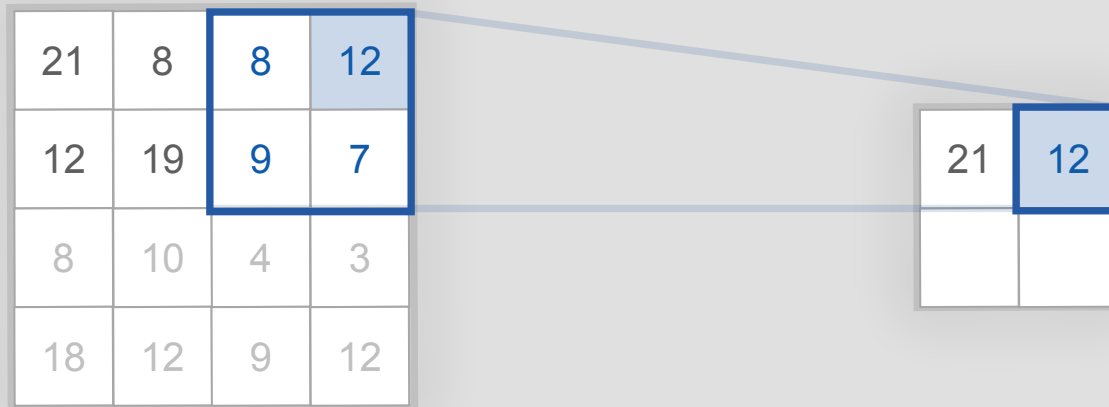
Pooling layer

- Pooling aggregates a fragment of feature map into a scalar
- Aggregating function can be different
- Usually, **max pooling** is applied



Pooling layer

- Pooling aggregates a fragment of feature map into a scalar
- Aggregating function can be different
- Usually, **max pooling** is applied



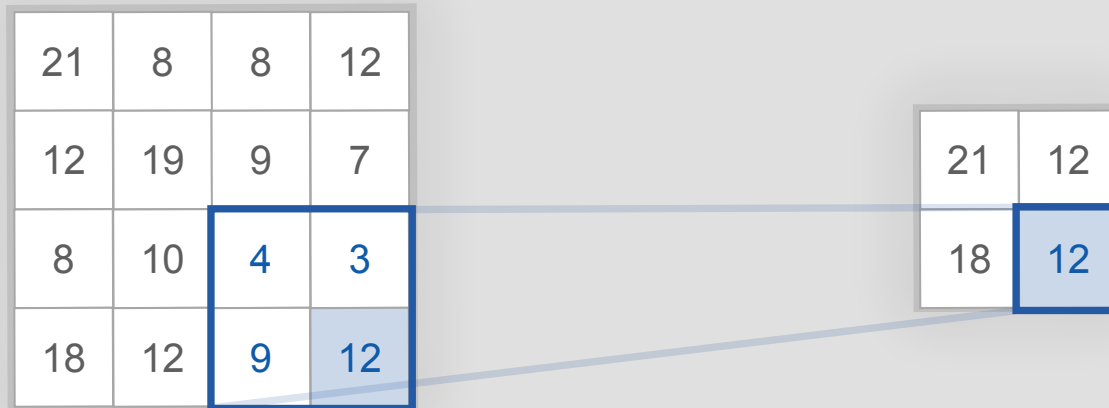
Pooling layer

- Pooling aggregates a fragment of feature map into a scalar
- Aggregating function can be different
- Usually, **max pooling** is applied



Pooling layer

- Pooling aggregates a fragment of feature map into a scalar
- Aggregating function can be different
- Usually, **max pooling** is applied



CNN for texts

- Usually, shallow networks with one convolutional layer and one pooling are used
- Input is a matrix of word embeddings



- Yoon Kim. *Convolutional Neural Networks for Sentence Classification* // EMNLP. — 2014.



CNN for texts

- Usually, shallow networks with one convolutional layer and one pooling are used
- Input is a matrix of word embeddings

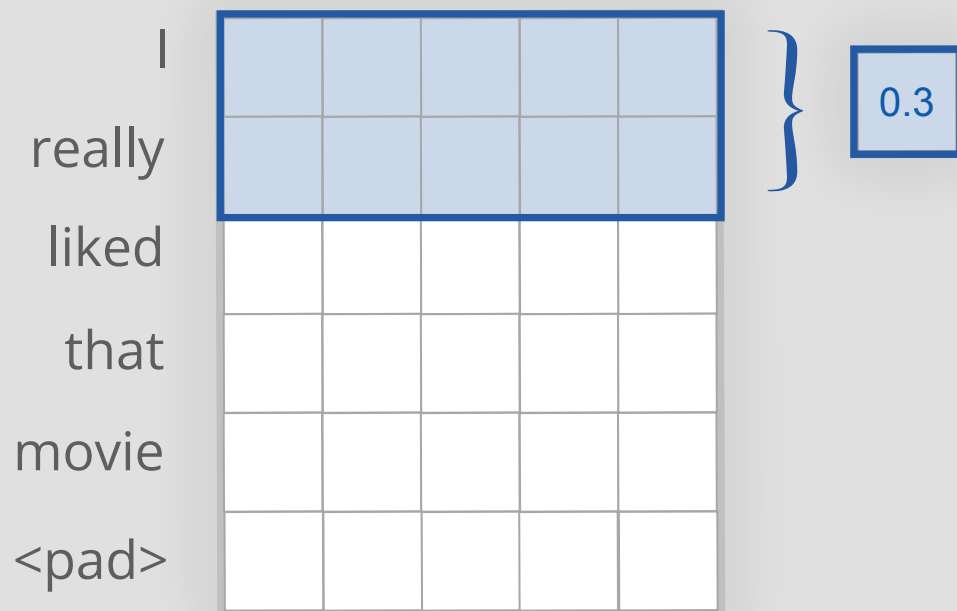
- Input is padded to fixed length
- **<pad> token embedding** must not affect the pooling!

I
really
liked
that
movie
<pad>



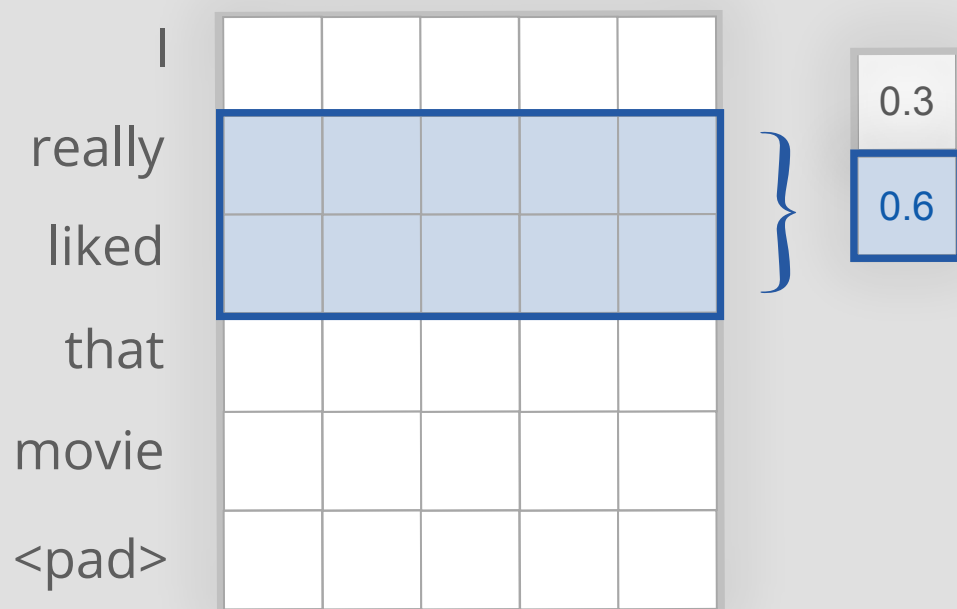
CNN for texts

- One-dimensional filters are used
- First dimension k corresponds to a number of words in the local context
- Second dimension is fixed and equal to word embeddings length



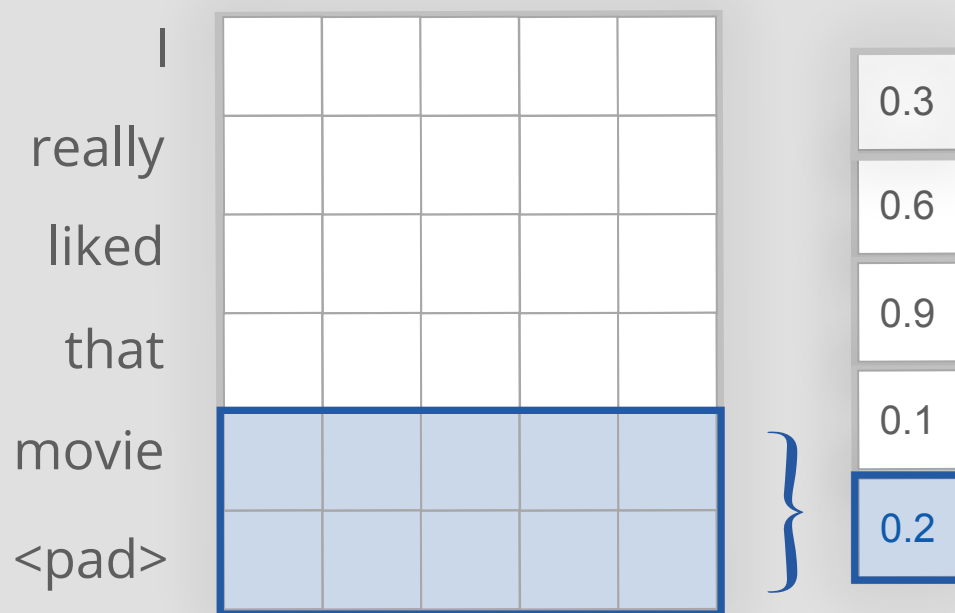
CNN for texts

- One-dimensional filters are used
- First dimension k corresponds to a number of words in the local context
- Second dimension is fixed and equal to word embeddings length



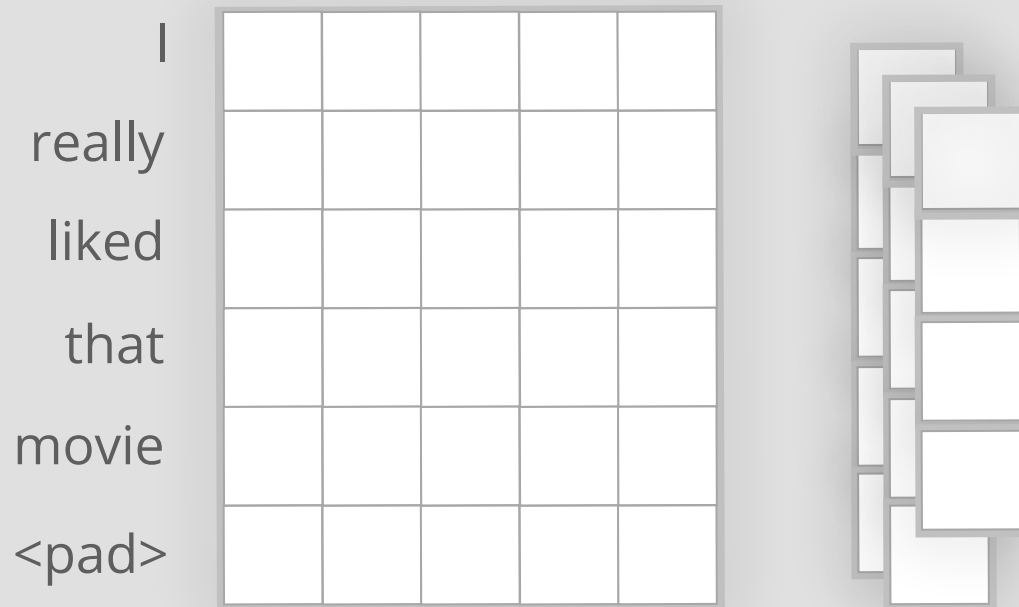
CNN for texts

- One-dimensional filters are used
- First dimension k corresponds to a number of words in the local context
- Second dimension is fixed and equal to word embeddings length



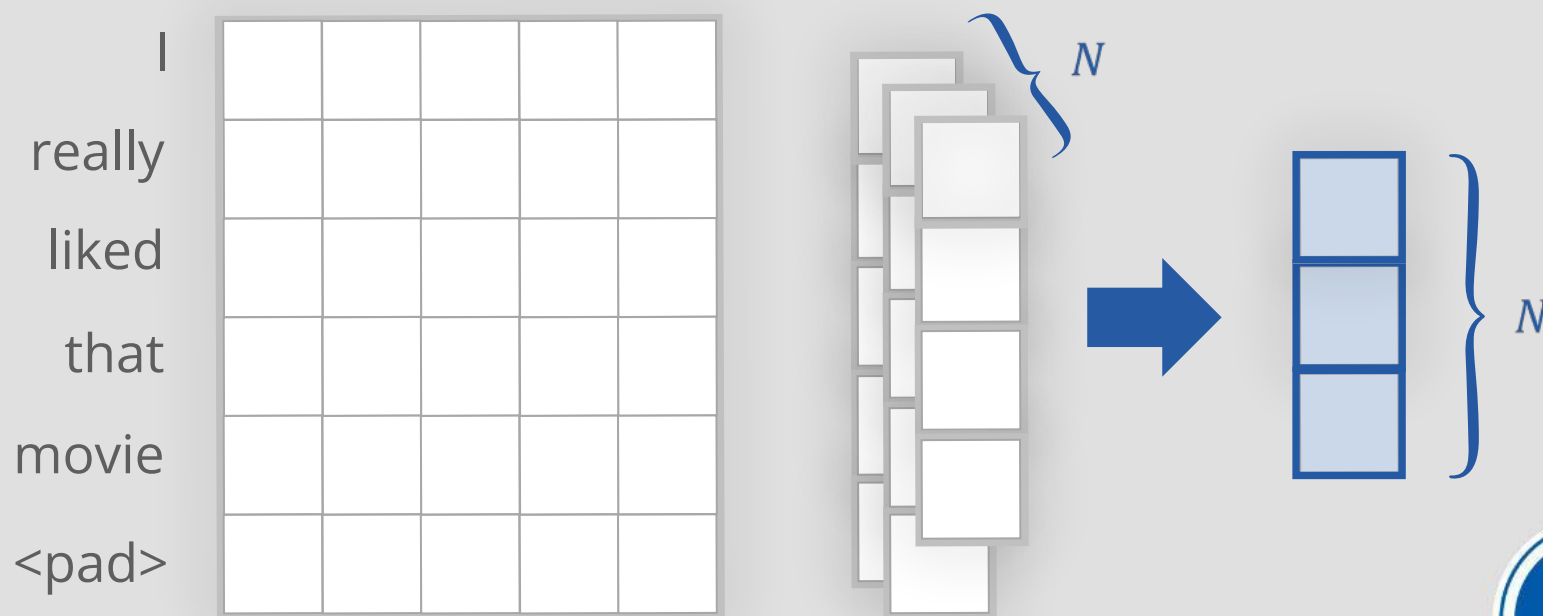
CNN for texts

- One-dimensional filters are used
- There can be several filters of different lengths



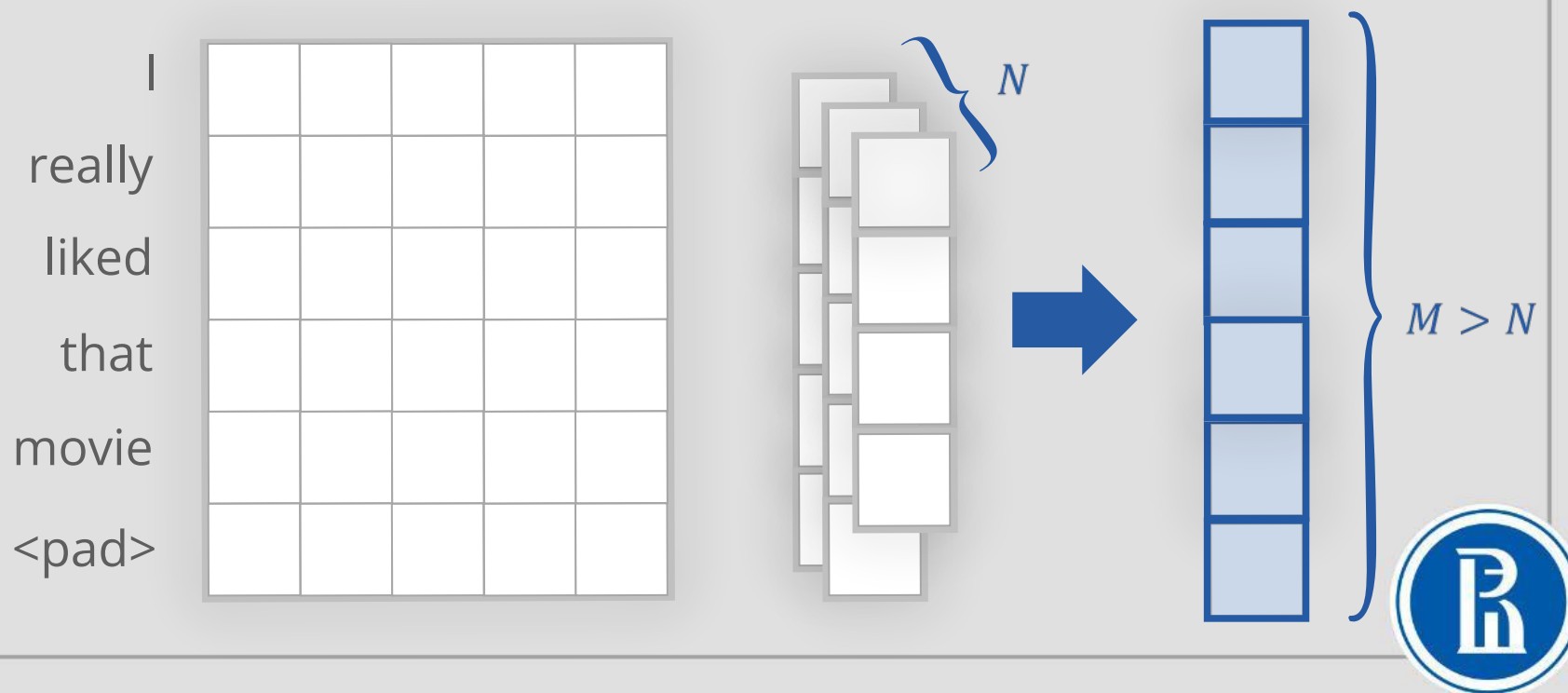
CNN for texts

- One-dimensional filters are used
- There can be several filters of different lengths
- Then we use **max-over-time pooling** instead of pooling
- It chooses **one feature** from each map



CNN for texts

- One-dimensional filters are used
- There can be several filters of different lengths
- **k-max** pooling can be applied



Input embeddings

- Word embeddings can be trained in the first layer from scratch
- Pretrained [word2vec](#) or [GloVe](#) embeddings can be used
- Embeddings can be both frozen and tuned during network training



Input embeddings

- Word embeddings can be trained in the first layer from scratch
- Pretrained [word2vec](#) or [GloVe](#) embeddings can be used
- Embeddings can be both frozen and tuned during network training
- Several [input channels](#) can be used
- For example, both channels have input [word2vec](#) embeddings, but the first one is frozen and the second is tuned
- Convolution is applied to both inputs, the result can be summarized



What's next

- One-hot character embeddings can be used as input
- The number of convolutional and pooling layers can be increased
- It is useful to add non-linear layers
- Dropout can be added after fully connected layers

- *Xiang Zhang, Junbo Zhao, Yann LeCun. Character-level Convolutional Networks for Text Classification // NIPS. — 2015.*



Main conclusions

- Convolutional networks originally designed for image processing work good with texts
- Input data is a matrix of word or character embeddings
- Filters of different sizes and max-over-time pooling can be used
- In classification problems, CNN allow to take into account the local context of words without explicitly using dictionary N-grams



Data processing in text classification



Classification solution steps

- Choosing the right quality metric
- Training data collection
- Data preprocessing
- Feature collection
- Choosing the model
- Model tuning



Data collection

- Even raw (unlabeled) texts can be difficult to collect
- If there is data in open sources, you can use [web-crawling](#):
 - Some websites have public API's
 - If they don't, you can use web-scraping (for example, [Scrapy](#) package for Python)



Data collection

- Even raw (unlabeled) texts can be difficult to collect
- If there is data in open sources, you can use [web-crawling](#):
- Some websites have public API's
- If they don't, you can use web-scraping (for example, [Scrapy](#) package for Python)
- Sometimes proper data doesn't exist at all
- You can use the power of [crowdsourcing](#):
 - [Yandex Toloka](#)
 - [Amazon Mechanical Turk](#)
 - ...



Data markup

- Unlabeled texts are useful as is
- But classification problem requires labeled data
- In practice, marking up small samples is often done manually by yourself
- Sometimes part of the documents can be marked up by defining simple heuristic rules:
 - By particular words inclusion
 - By regular expression
- In all other cases you can use assessors (better), or crowdsourcing (faster)



Additional markup and active learning

- Models can be applied to data that changes dynamically over time
- **Examples:** news, memes
- Once trained, model can become obsolete with time
- Usually it is necessary to collect additional data and tune model on regular basis



Additional markup and active learning

- Models can be applied to data that changes dynamically over time
- **Examples:** news, memes
- Once trained, model can become obsolete with time
- Usually it is necessary to collect additional data and tune model on regular basis
- We can reduce the amount of additionally marked data by choosing samples that can provide useful information for the model
- **Idea:** send the samples **where the model is least confident**

