Обработка текстов с помощью нейросетей

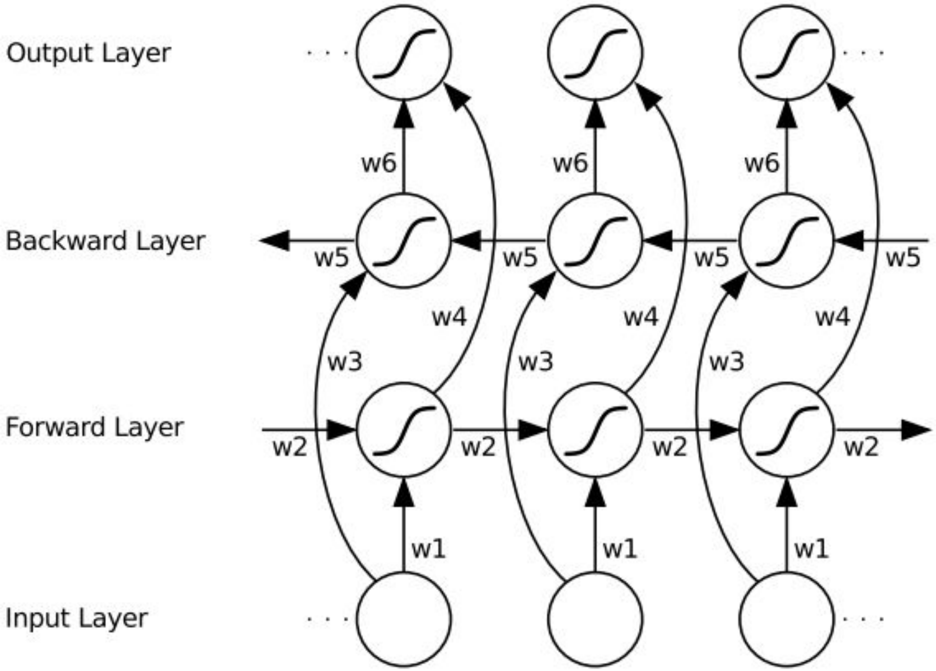
Grigory Sapunov CTO / Intento

HSE / #10 / 26.11.2019

Advanced RNNs

Bidirectional RNN (BRNN/BLSTM)

Часто последовательность доступна сразу целиком, так что её можно сканировать в обоих направлениях.



Typical result: BRNN>RNN, LSTM>RNN, BLSTM>BRNN

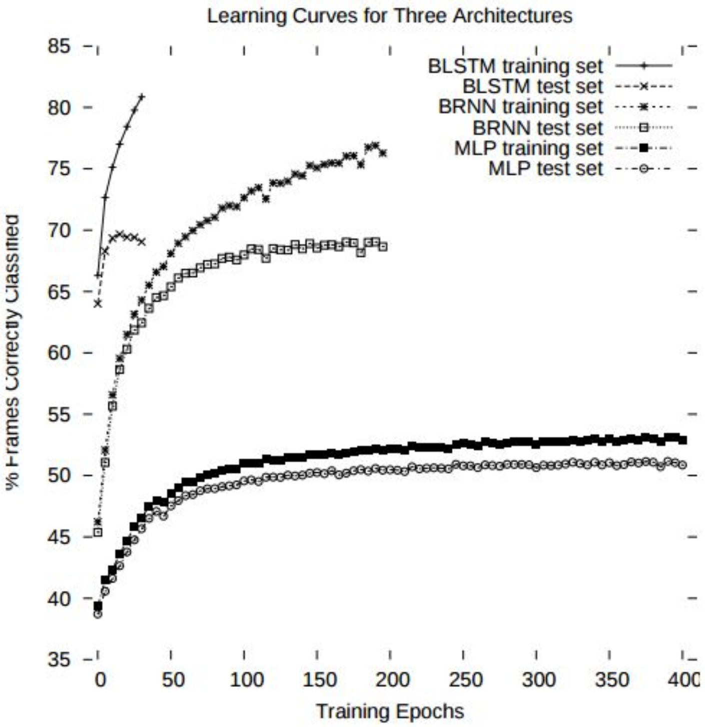
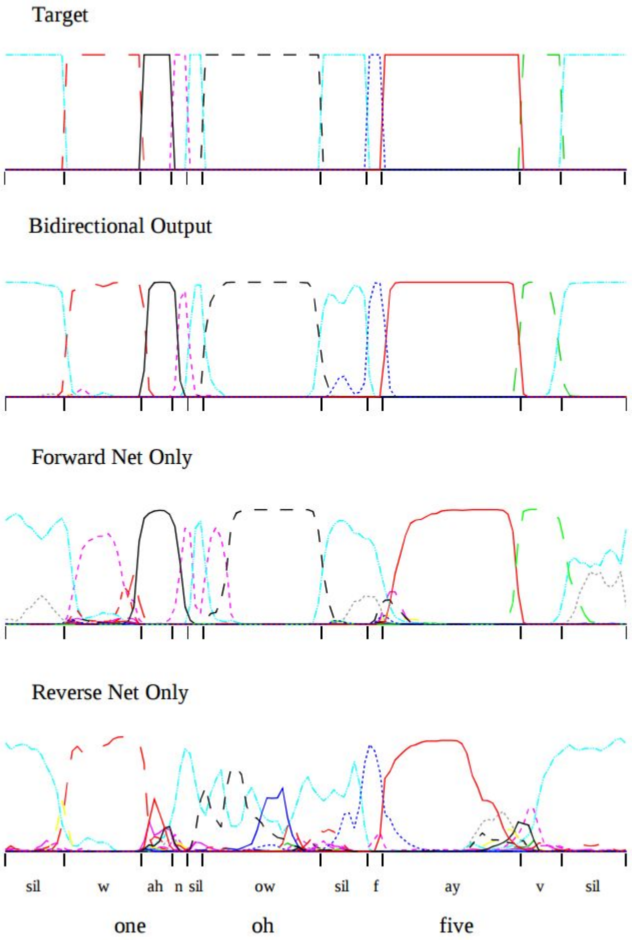


Figure 5.3: **Learning curves on TIMIT for BLSTM, BRNN and MLP with no time-window.** For all experiments, LSTM was much faster to converge than either the RNN or MLP architectures.

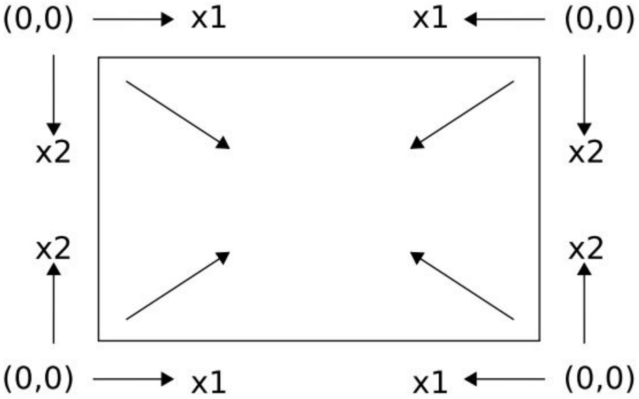
Example: BLSTM classifying the utterance “one oh five”

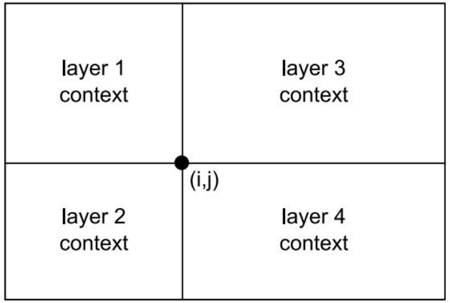


Multidimensional and Multidirectional RNNs

RNN могут также быть многомерными и многонаправленными.

Это более естественно для многомерных данных, например, изображений.





Multidirectional Multidimensional RNN/LSTM

Standard RNNs are inherently one dimensional, and therefore poorly suited to multidimensional data (e.g. images).

The basic idea of MDRNNs (Graves, Fernandez, Schmidhuber, 2007) is to replace the single recurrent connection found in standard RNNs with as many recurrent connections as there are dimensions in the data.

It assumes some ordering on the multidimensional data. BRNNs can be extended to n-dimensional data by using 2^n separate hidden layers.

Multi-dimensionality (MDRNN)

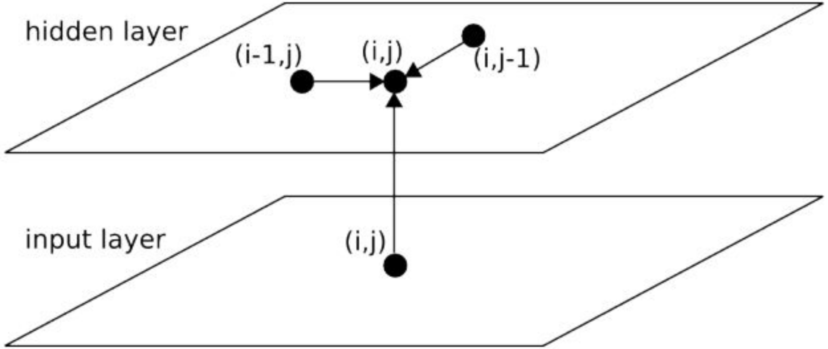


Figure 8.1: **MDRNN forward pass**

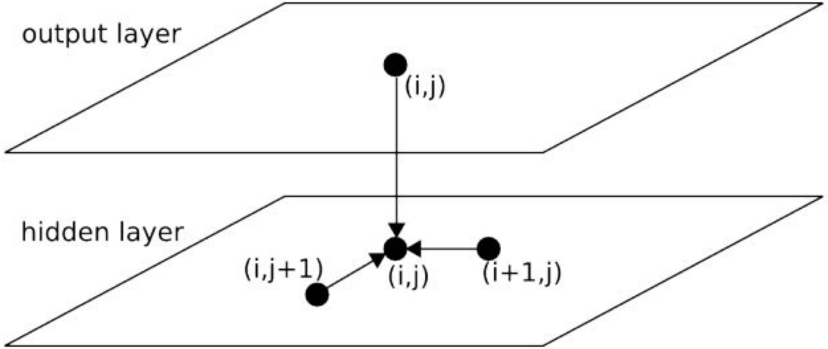


Figure 8.2: **MDRNN backward pass**

The basic idea of MDRNNs is to replace the single recurrent connection found in standard RNNs with as many recurrent connections as there are dimensions in the data.

Multi-directionality (MDMDRNN?)

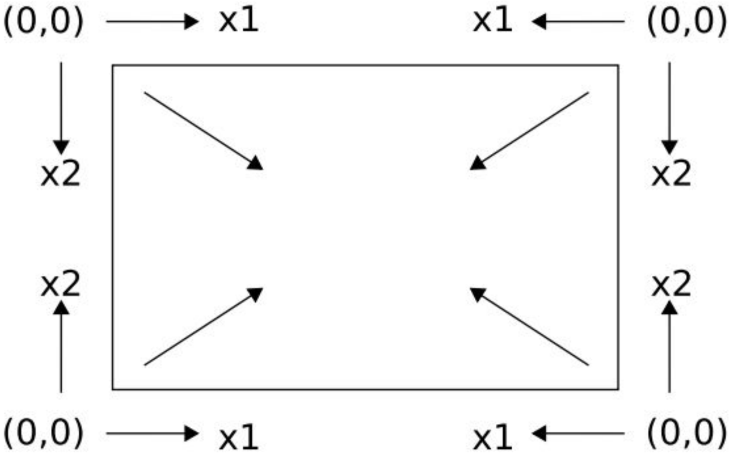


Figure 8.5: **Axes used by the 4 hidden layers in a multidirectional MDRNN.** The arrows inside the rectangle indicate the direction of propagation during the forward pass.

Multidirectional multidimensional RNN (MDMDRNN?)

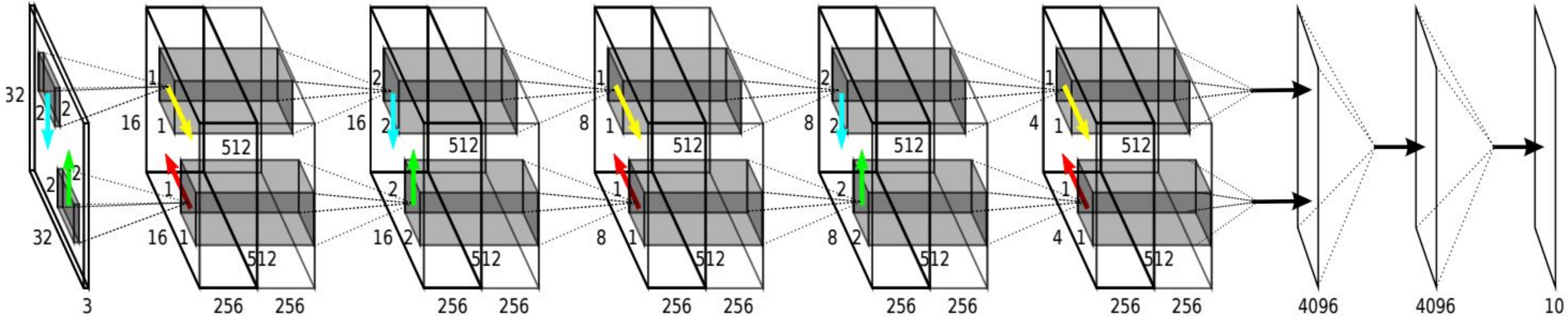
The previously mentioned ordering is not the only possible one. It might be OK for some tasks, but it is usually preferable for the network to have access to the surrounding context in all directions. This is particularly true for tasks where precise localisation is required, such as image segmentation.

For one dimensional RNNs, the problem of multidirectional context was solved by the introduction of bidirectional recurrent neural networks (BRNNs). BRNNs contain two separate hidden layers that process the input sequence in the forward and reverse directions.

BRNNs can be extended to n-dimensional data by using 2^n separate hidden layers, each of which processes the sequence using the ordering defined above, but with a different choice of axes.

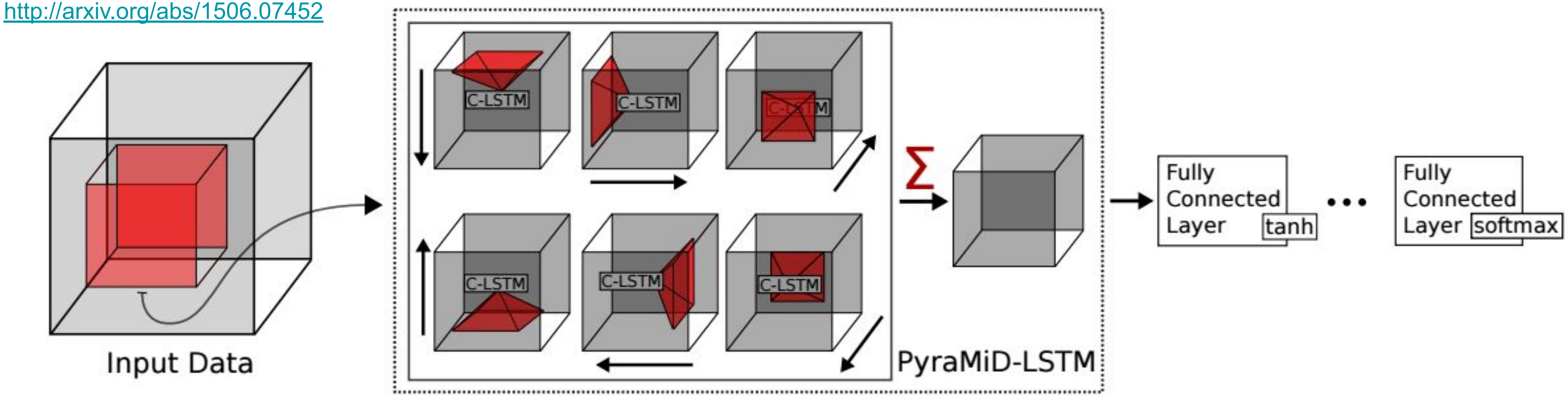
ReNet (2015)

http://arxiv.org/abs/1505.00393



PyraMiD-LSTM (2015)

http://arxiv.org/abs/1506.07452



Tree-LSTM (2015)

Interesting LSTM generalisation: Tree-LSTM

“However, natural language exhibits syntactic properties that would naturally combine words to phrases. We introduce the Tree-LSTM, a generalization of LSTMs to tree-structured network topologies. Tree-LSTMs outperform all existing systems and strong LSTM baselines on two tasks: predicting the semantic relatedness of two sentences and sentiment classification.”

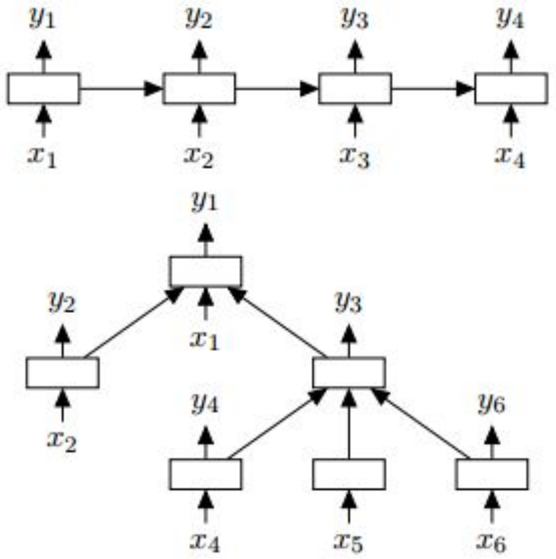


Figure 1: **Top:** A chain-structured LSTM network. **Bottom:** A tree-structured LSTM network with arbitrary branching factor.

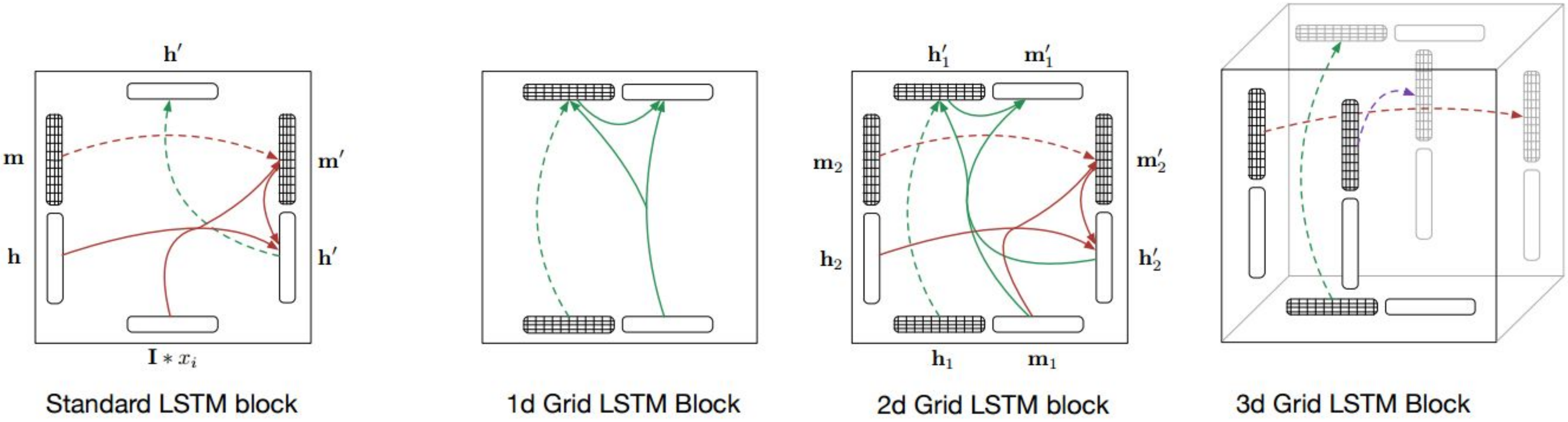
Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks, https://arxiv.org/abs/1507.01526

Grid LSTM (2016)

Another interesting LSTM generalisation: Grid LSTM

“This paper introduces Grid Long Short-Term Memory, a network of LSTM cells **arranged in a multidimensional grid** that can be applied to vectors, sequences or higher dimensional data such as images. The network differs from existing deep LSTM architectures in that the cells are connected between network layers as well as along the spatiotemporal dimensions of the data. The network provides a unified way of using LSTM for both deep and sequential computation.”

Grid LSTM (2016)



Standard LSTM block

1d Grid LSTM Block

2d Grid LSTM block

3d Grid LSTM Block

Figure 1: Blocks form the standard LSTM and those that form Grid LSTM networks of $N = 1, 2$ and 3 dimensions. The dashed lines indicate identity transformations. The standard LSTM block does not have a memory vector in the vertical dimension; by contrast, the 2d Grid LSTM block has the memory vector $\mathbf{m}_1$ applied along the vertical dimension.

https://arxiv.org/abs/1507.01526

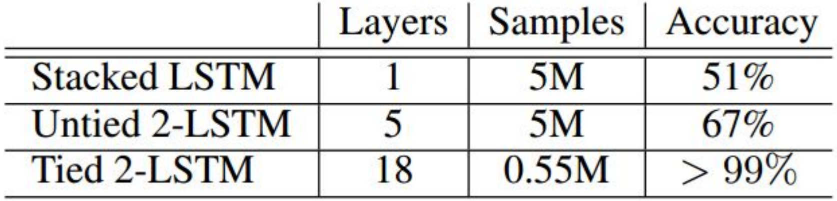
Grid LSTM (2016)

One-dimensional Grid LSTM corresponds to a feed-forward network that uses LSTM cells in place of transfer functions such as tanh and ReLU. These networks are related to Highway Networks (Srivastava et al., 2015) where a gated transfer function is used to successfully train feed-forward networks with up to 900 layers of depth.

Grid LSTM with **two dimensions** is analogous to the Stacked LSTM, but it adds cells along the depth dimension too.

Grid LSTM with **three or more dimensions** is analogous to Multidimensional LSTM, but differs from it not just by having the cells along the depth dimension, but also by using the proposed mechanism for modulating the N-way interaction that is not prone to the instability present in Multidimesional LSTM.

Grid LSTM (2016)



	Layers	Samples	Accuracy
Stacked LSTM	1	5M	51%
Untied 2-LSTM	5	5M	67%
Tied 2-LSTM	18	0.55M	$> 99\%$

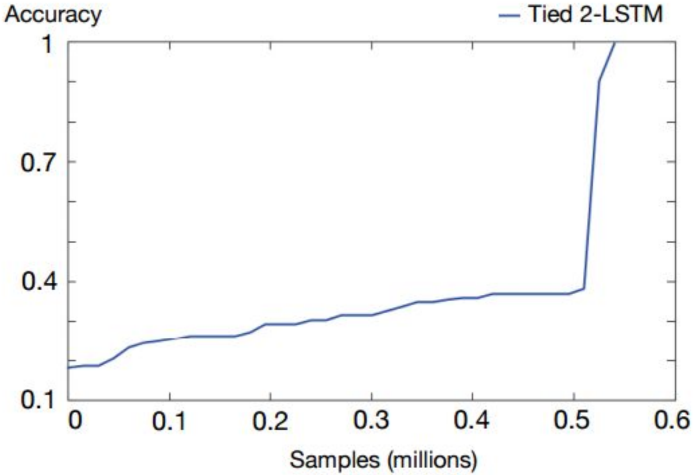


Figure 4: Results on 15-digit addition. The left table gives results for the best performing networks of each type. The right graph depicts the learning curve of the 18-layer tied 2-LSTM that solves the problem with less than 550K examples. The spike in the curve is likely due to the repetitions in the steps of the addition algorithm.

Работа с текстом

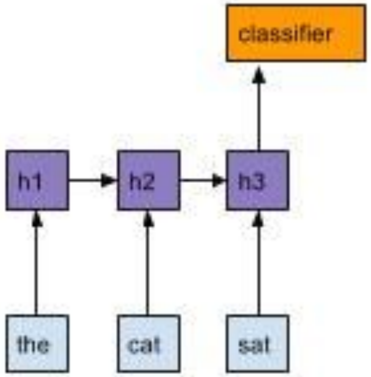
Векторизация (Vectorizing)

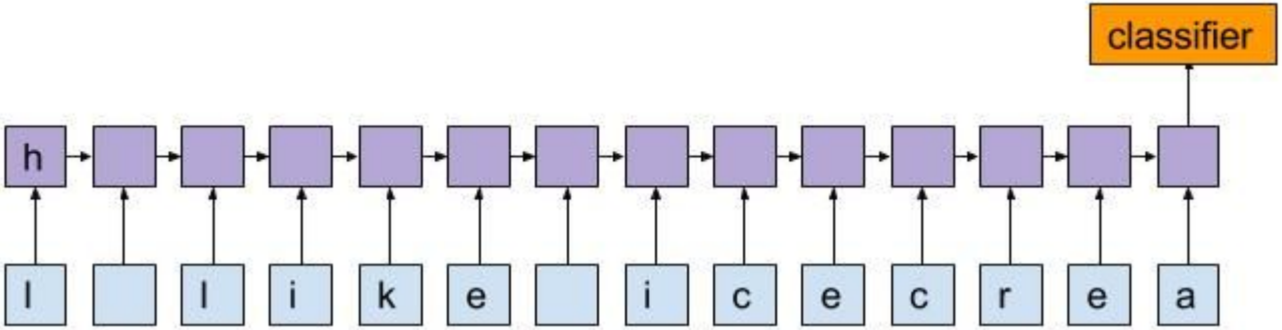
Векторизация — процесс превращения текста в тензор с числами.

Может быть реализована различными способами:

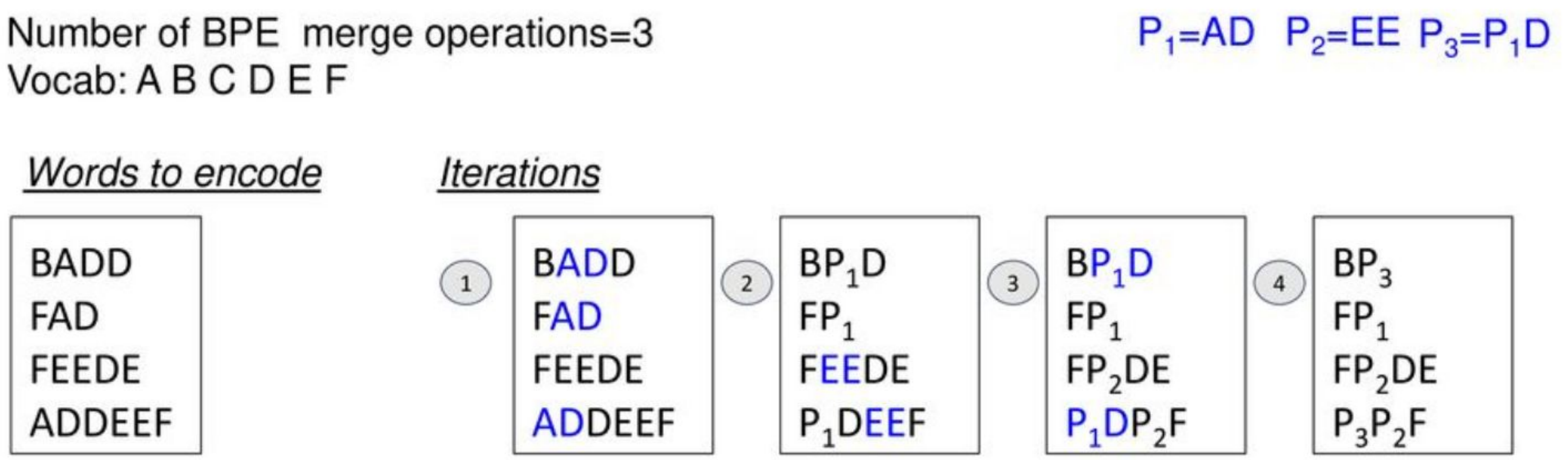
- Сегментация текста на **слова** и преобразование каждого слова в вектор.
- Сегментация текста на **символы** и преобразование каждого символа в вектор.
- Выделение **n-грамм** слов или символов и преобразование каждой n-граммы в вектор.
- **Subwords embeddings** на основе **Byte-Pair Encoding (BPE)**: итеративно заменяем наиболее частую пару символов на новый символ.

Word-level vs. Character-level





BPE Example



Number of BPE merge operations=3
Vocab: A B C D E F

P_1=AD P_2=EE P_3=P_1D

Words to encode

BADD
FAD
FEEDE
ADDEEF

Iterations

1

BADD
FAD
FEEDE
ADDEEF

2

BP_1D
FP_1
FEEDE
P_1DEEF

3

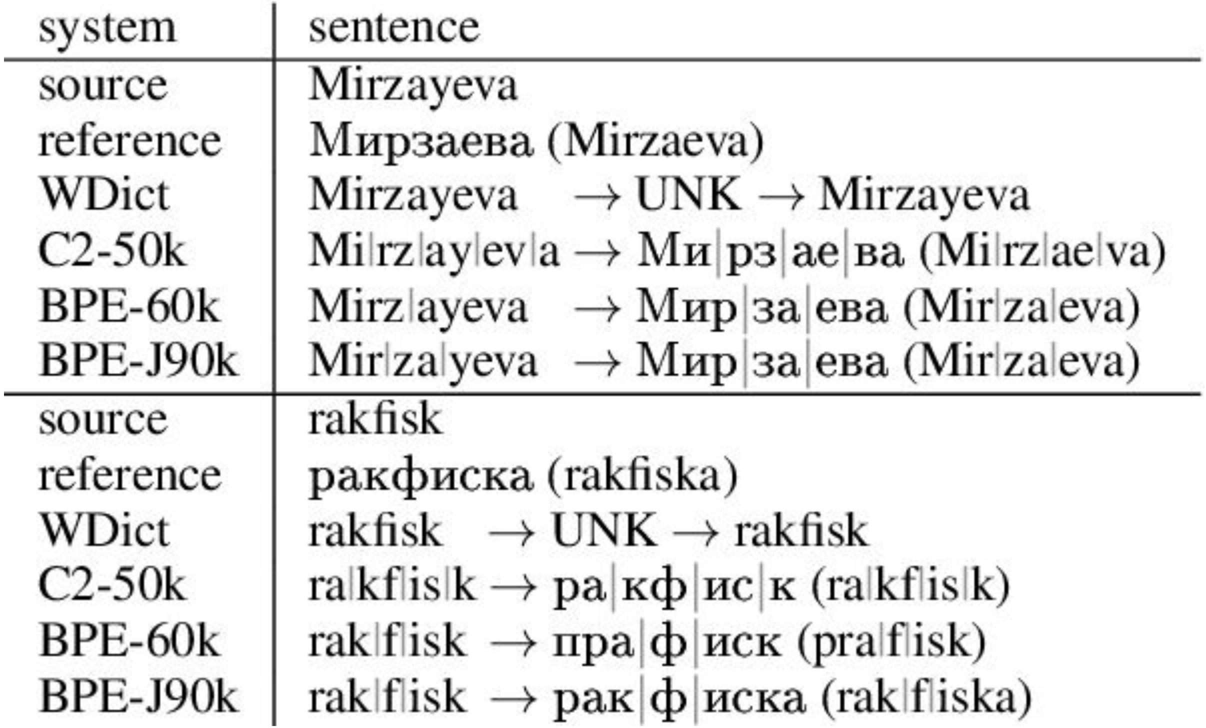
BP_1D
FP_1
FP_2DE
P_1DP_2F

4

BP_3
FP_1
FP_2DE
$P_3$$P_2$F

https://slideplayer.com/slide/14627970/

BPE Example



system	sentence
source	Mirzayeva
reference	Мирзаева (Mirzaeva)
WDict	Mirzayeva → UNK → Mirzayeva
C2-50k	Mi|rz|ay|ev|a → Ми|рз|ае|ва (Mi|rz|ae|va)
BPE-60k	Mirz|ayeva → Мир|за|ева (Mir|za|eva)
BPE-J90k	Mir|za|yeva → Мир|за|ева (Mir|za|eva)
source	rakfisk
reference	ракфиска (rakfiska)
WDict	rakfisk → UNK → rakfisk
C2-50k	ra|kf|is|k → ра|кф|ис|к (ra|kf|is|k)
BPE-60k	rak|f|isk → пра|ф|иск (pra|f|isk)
BPE-J90k	rak|f|isk → рак|ф|иска (rak|f|iska)

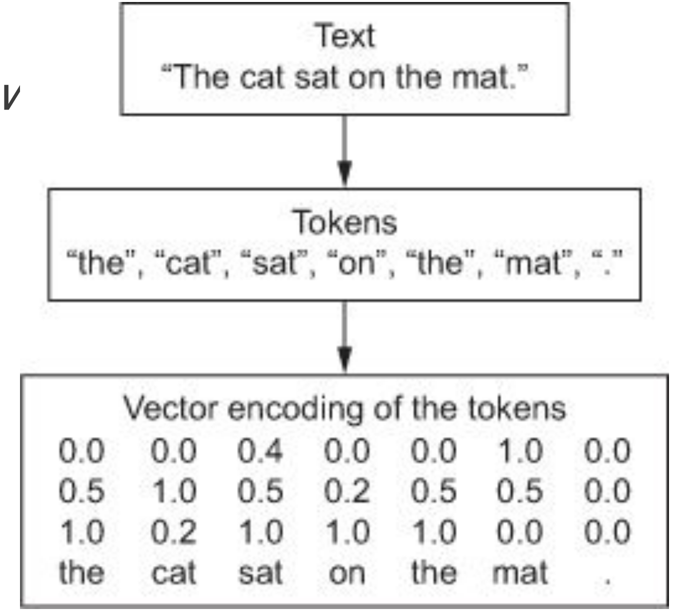
https://arxiv.org/abs/1508.07909

Векторизация (Vectorizing)

Элементы, на которые разбивается текст, называются токенами, а процесс разбиения на токены — **токенизацией**.

Преобразование токена в вектор может происходи

- *one-hot encoding*
- *token embedding* (*word embedding* для слов).



Text
“The cat sat on the mat.”

Tokens
“the”, “cat”, “sat”, “on”, “the”, “mat”, “.”

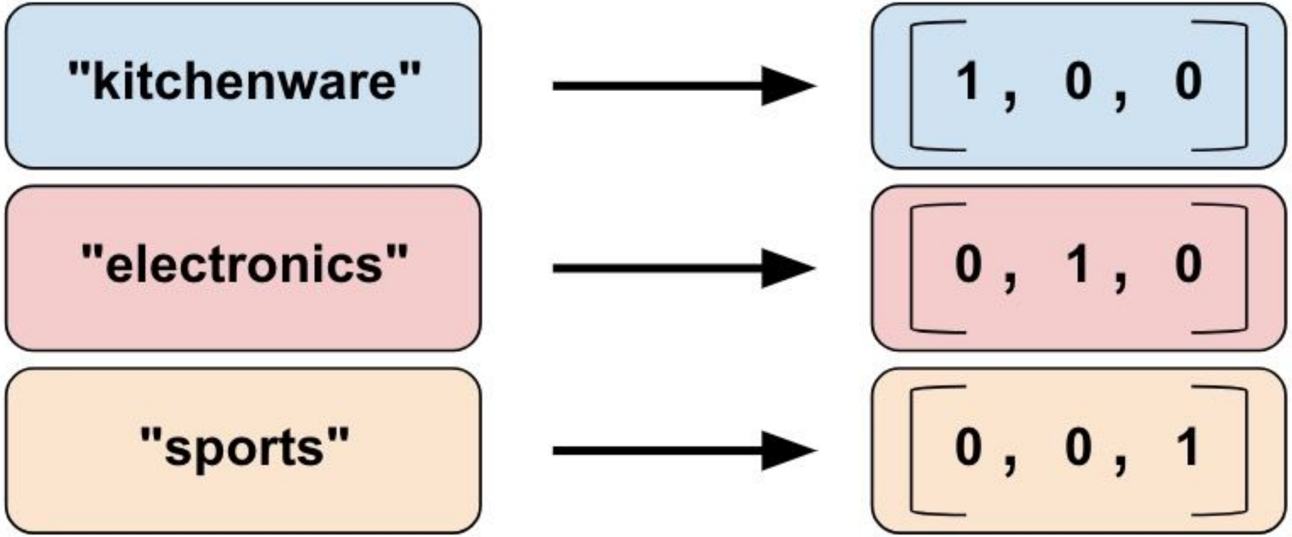
Vector encoding of the tokens

0.0	0.0	0.4	0.0	0.0	1.0	0.0
0.5	1.0	0.5	0.2	0.5	0.5	0.0
1.0	0.2	1.0	1.0	1.0	0.0	0.0
the	cat	sat	on	the	mat	.

https://www.manning.com/books/deep-learning-with-python

One-hot encoding

- **Word-level** one-hot encoding
- **Character-level** one-hot encoding



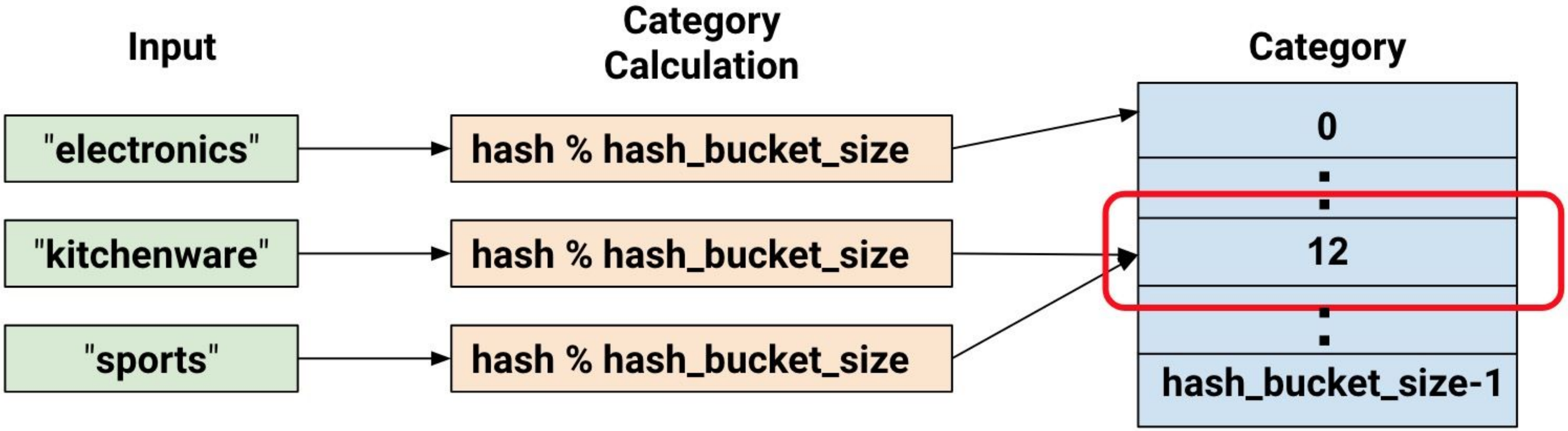
"kitchenware" → [1 , 0 , 0]

"electronics" → [0 , 1 , 0]

"sports" → [0 , 0 , 1]

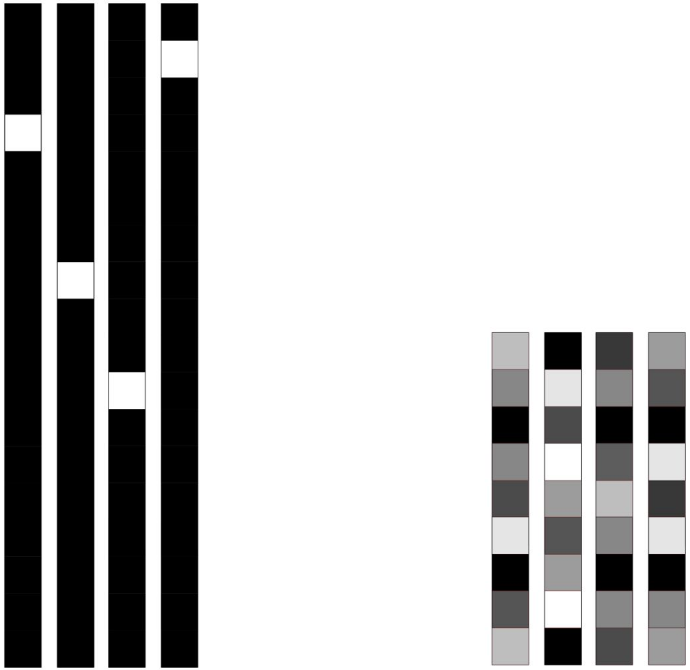
One-hot hashing trick

Удобен при большом словаре



https://www.tensorflow.org/guide/feature_columns

Word embeddings



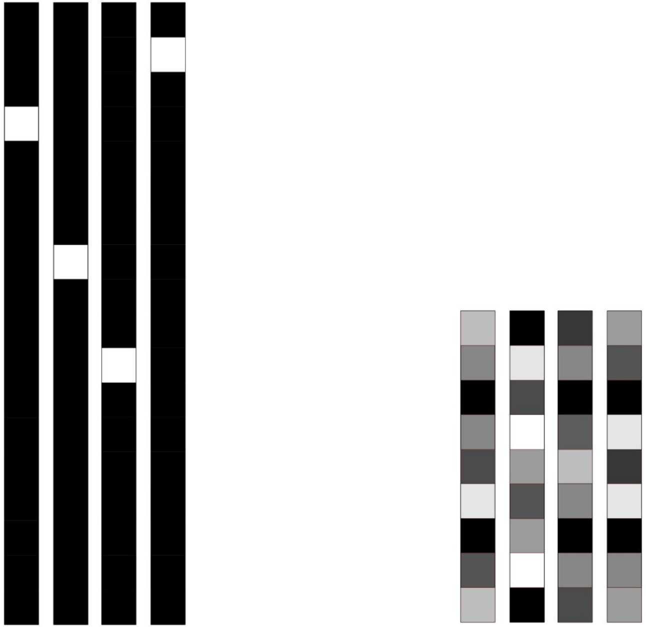
One-hot word vectors:
- Sparse
- High-dimensional
- Hard-coded

Word embeddings:
- Dense
- Lower-dimensional
- Learned from data

Word embeddings

Наивный вариант: использовать случайные векторы для слов.

Проблема: похожие слова имеют сильно различающиеся вектора.

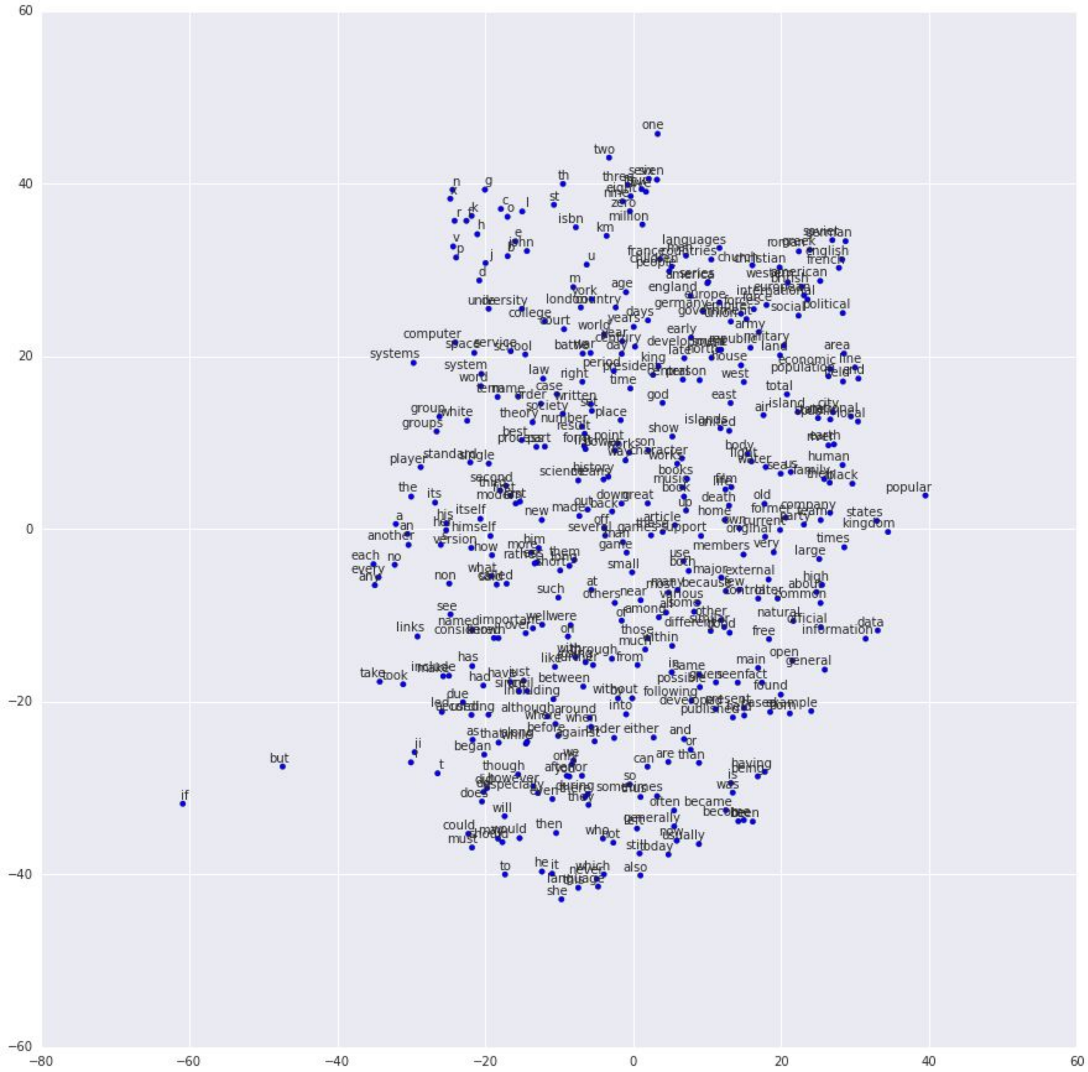


One-hot word vectors:
- Sparse
- High-dimensional
- Hard-coded

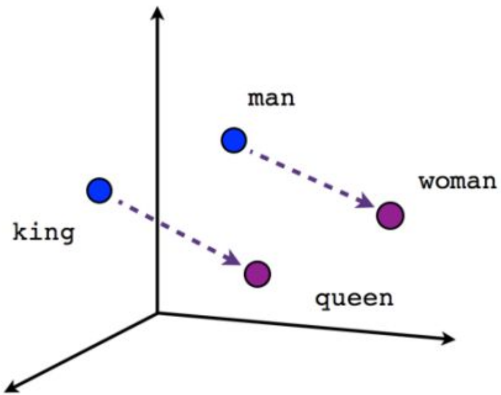
Word embeddings:
- Dense
- Lower-dimensional
- Learned from data

Word embeddings

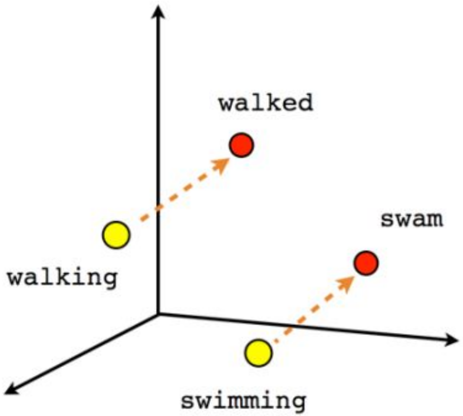
Хотим, чтобы структура этого пространства отражала отношения между словами, а расстояние в этом пространстве соответствовало семантическому расстоянию между словами.



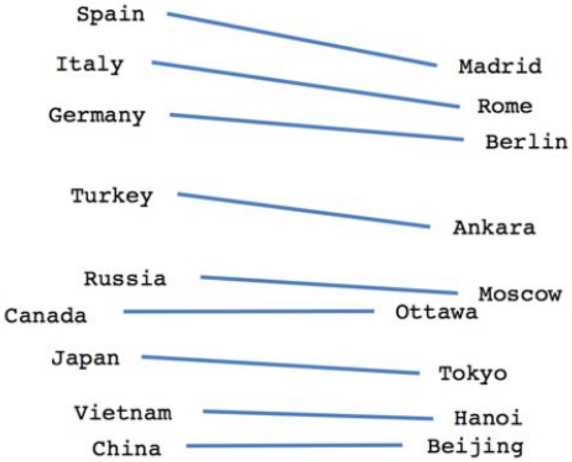
Word embeddings



Male-Female



Verb tense

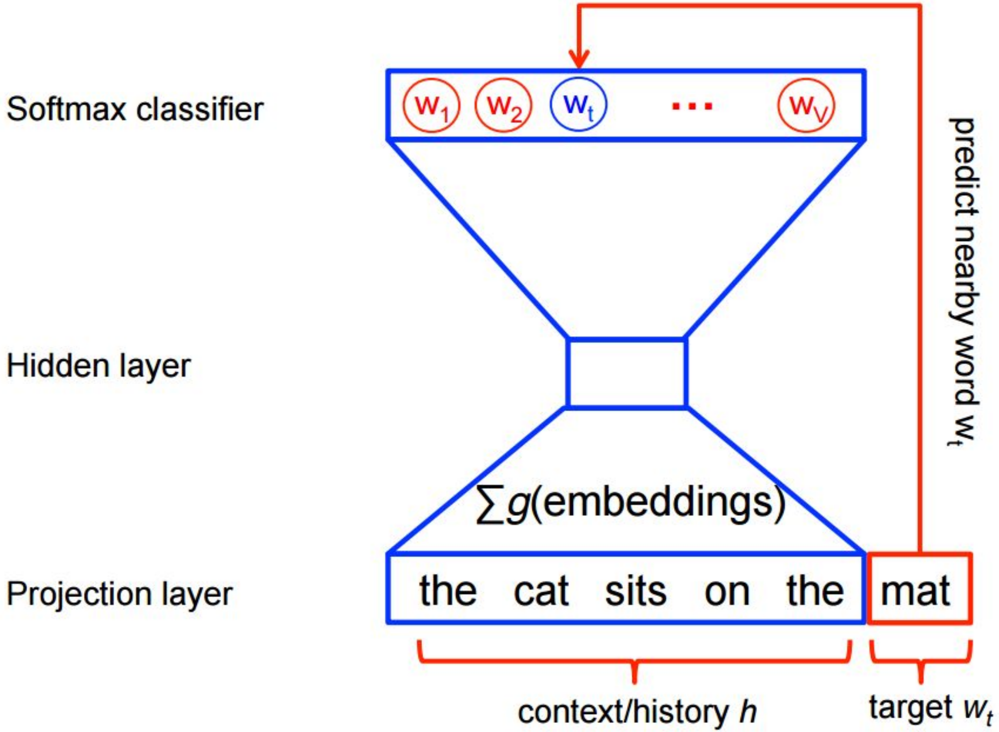


Country-Capital

https://www.tensorflow.org/tutorials/representation/word2vec

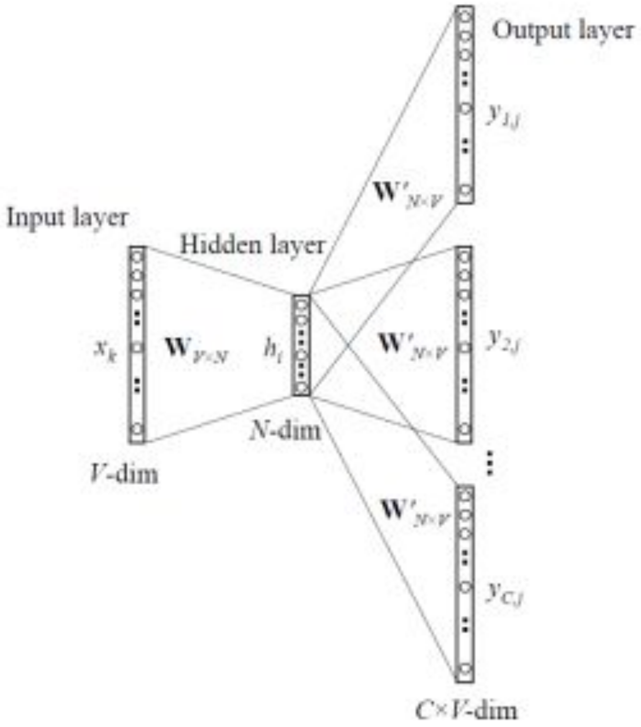
Word embeddings

Distributional Hypothesis states that words that appear in the same contexts share semantic meaning

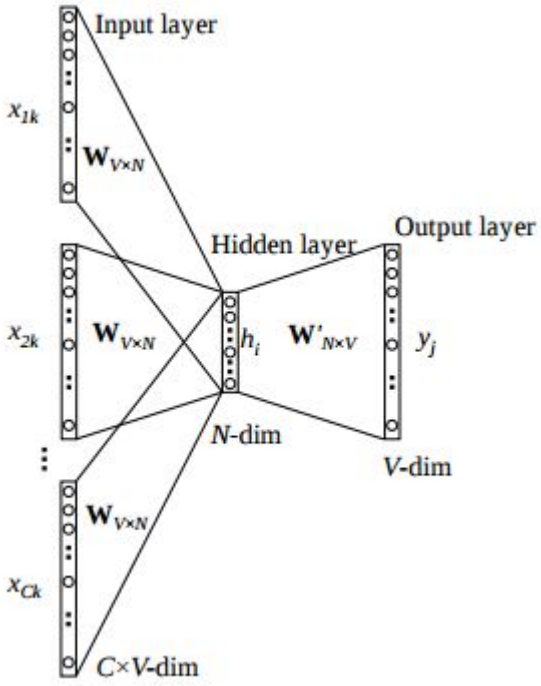


https://www.tensorflow.org/tutorials/representation/word2vec

Word2vec



Skip-gram model



Continuous Bag of Words (CBOW)

https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c

Word2vec

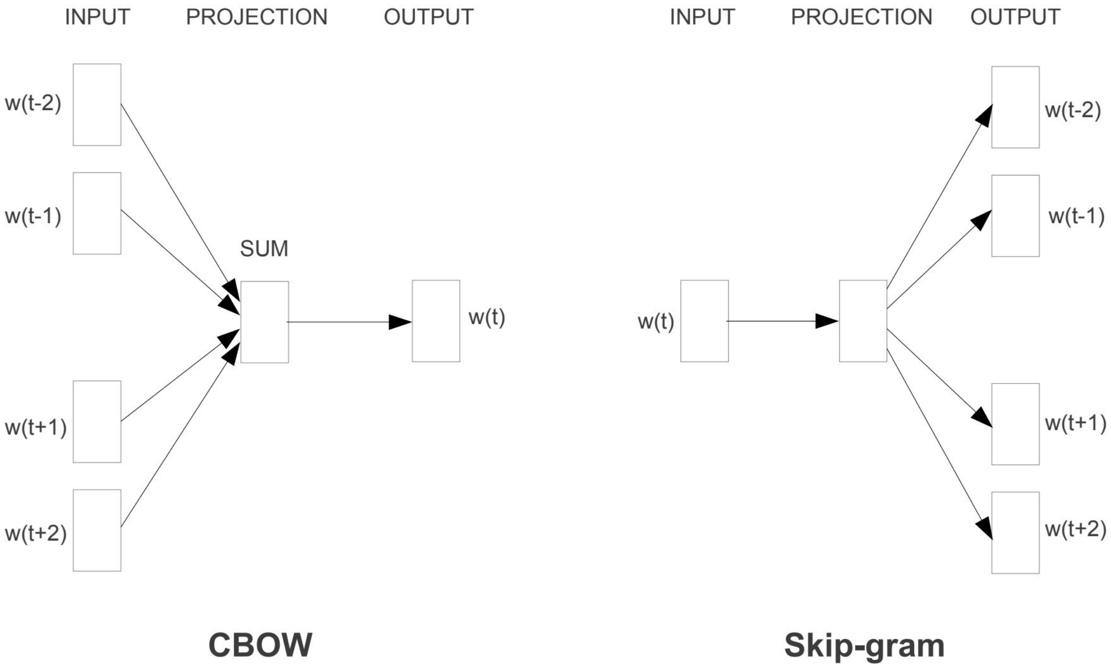


Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

https://arxiv.org/pdf/1301.3781.pdf

Получение word embeddings

- Выучить из данных
  - https://radimrehurek.com/gensim/models/word2vec.html
  - https://radimrehurek.com/gensim/models/fasttext.html#gensim.models.fasttext.FastText
  - https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/
  - https://keras.io/layers/embeddings/
  - https://github.com/google/sentencepiece (BPE)
- Использовать пре-тренированные (Word2vec, GloVe, FastText, BPE)
  - http://mccormickml.com/2016/04/12/googles-pretrained-word2vec-model-in-python/
  - https://blog.manash.me/how-to-use-pre-trained-word-vectors-from-facebooks-fasttext-a71e6d55f27
  - https://github.com/bheinzerling/bpemb (BPE)
- Использовать transfer learning (дообучить пре-тренированные)
  - https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html

Embedding layer

Turns positive integers (indexes) into dense vectors of fixed size.
eg. [[4], [20]] -> [[0.25, 0.1], [0.6, -0.2]]

It’s effectively a dictionary lookup:

Word index → Embedding Layer → Word vector

Input shape
2D tensor with shape: (batch_size, sequence_length).

Output shape
3D tensor with shape: (batch_size, sequence_length, output_dim).

https://keras.io/layers/embeddings/

Resources

- Tutorial on Word Embeddings: https://www.tensorflow.org/tutorials/text/word_embeddings
- Embedding layer: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Embedding
- More on BPE https://leimao.github.io/blog/Byte-Pair-Encoding/
- Ещё про embeddings https://towardsdatascience.com/beyond-word-embeddings-part-2-word-vectors-nlp-modeling-from-bow-to-bert-4ebd4711d0ec
- How to read: Character level deep learning https://offbit.github.io/how-to-read/ (a bit outdated, but ideas are useful)
- Word2vec tutorial https://www.kaggle.com/c/word2vec-nlp-tutorial

Практика (Colab):
Классификация текстов (IMDB)
с помощью разных сетей

IMDB Movie reviews sentiment classification

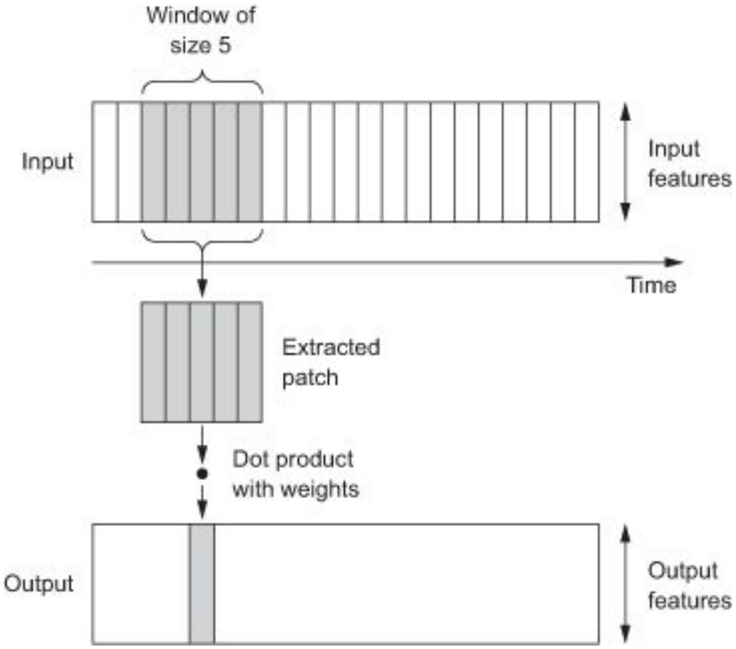
https://keras.io/datasets/

Dataset of 25,000 movies reviews from IMDB, labeled by sentiment (positive/negative). Reviews have been preprocessed, and each review is encoded as a sequence of word indexes (integers). For convenience, words are indexed by overall frequency in the dataset, so that for instance the integer "3" encodes the 3rd most frequent word in the data. This allows for quick filtering operations such as: "only consider the top 10,000 most common words, but eliminate the top 20 most common words".

As a convention, "0" does not stand for a specific word, but instead is used to encode any unknown word.

CNN for sequence classification

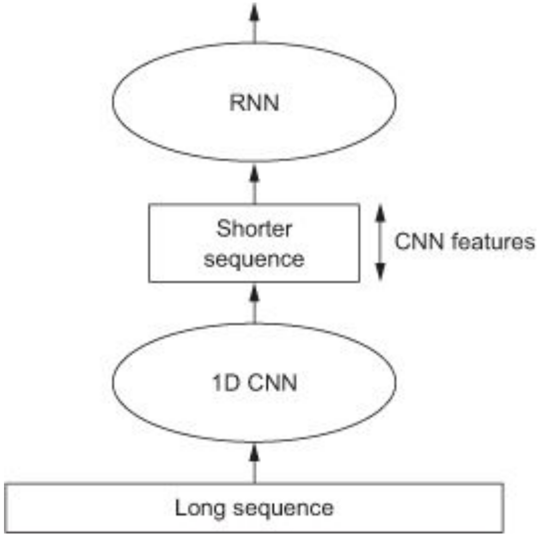
How 1D convolution works: each output timestep is obtained from a temporal patch in the input sequence.



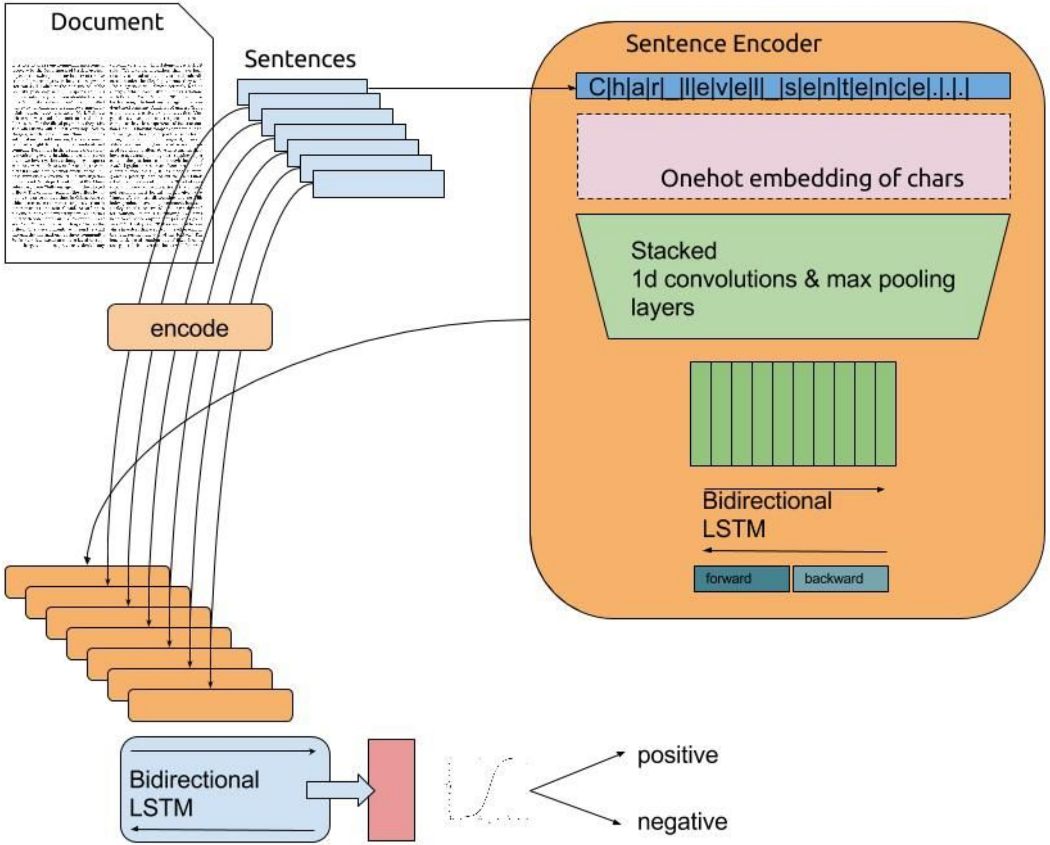
RNN + CNN

```
from keras.models import Sequential
from keras import layers
from keras.optimizers import RMSprop


model = Sequential()
model.add(layers.Conv1D(32, 5, activation='relu',
                        input_shape=(None, float_data.shape[-1])))
model.add(layers.MaxPooling1D(3))
model.add(layers.Conv1D(32, 5, activation='relu'))
model.add(layers.GRU(32, dropout=0.1, recurrent_dropout=0.5))
model.add(layers.Dense(1))
```



Сложный кейс: CNN + Hierarchy of LSTMs



https://offbit.github.io/how-to-read/

Resources

- Different approaches to text classification (RNN, Deep RNN, Bidirectional RNN, CNN, RNN+CNN): https://colab.research.google.com/drive/1uejcFOkuzzp97P-pwzICrWxrLJoTNpRv
- Earlier example on text classification using FFN (section called “MLP for binary classification”): https://colab.research.google.com/drive/1YVrUhphhL_CoTY-aNCJGJLIXmeJW_d62
- Text classification example from TF: https://www.tensorflow.org/tutorials/text/text_classification_rnn

H/W #2: Sentiment analysis на данных Twitter

Задание:

- Взять датасет с позитивными и негативными твитами:
  - http://study.mokoron.com/
  - Либо любой другой датасет на ваш выбор
- Подготовить датасет для использования нейросетью.
- Сформировать и обучить модель.
- Оценить качество на тестовой выборке.