Практика: Классификация текстов с помощью нейросети

Grigory Sapunov  CTO / Intento

HSE / #11 / 20.11.2018

Работа с текстом

Векторизация (Vectorizing)

Векторизация — процесс превращения текста в тензор с числами.

Может быть реализована различными способами:

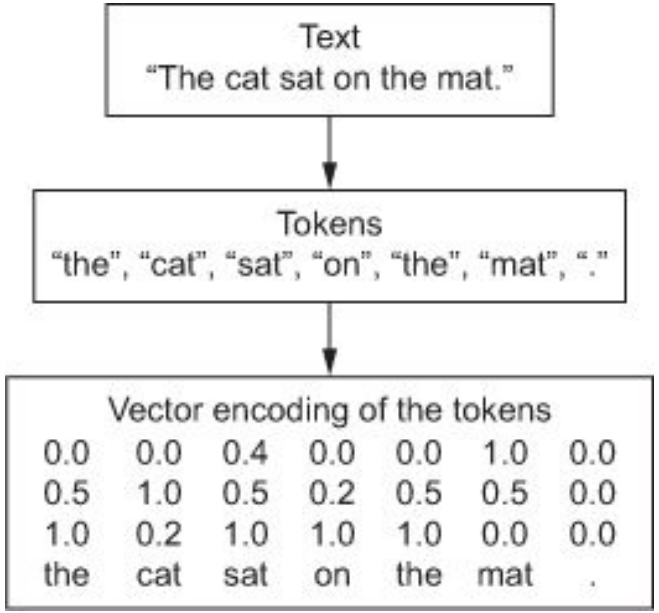
- Сегментация текста на **слова** и преобразование каждого слова в вектор.
- Сегментация текста на **символы** и преобразование каждого символа в вектор.
- Выделение **n-грамм** слов или символов и преобразование каждой n-граммы в вектор.

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

Векторизация (Vectorizing)

Преобразование токена в вектор может происходить разными способами:

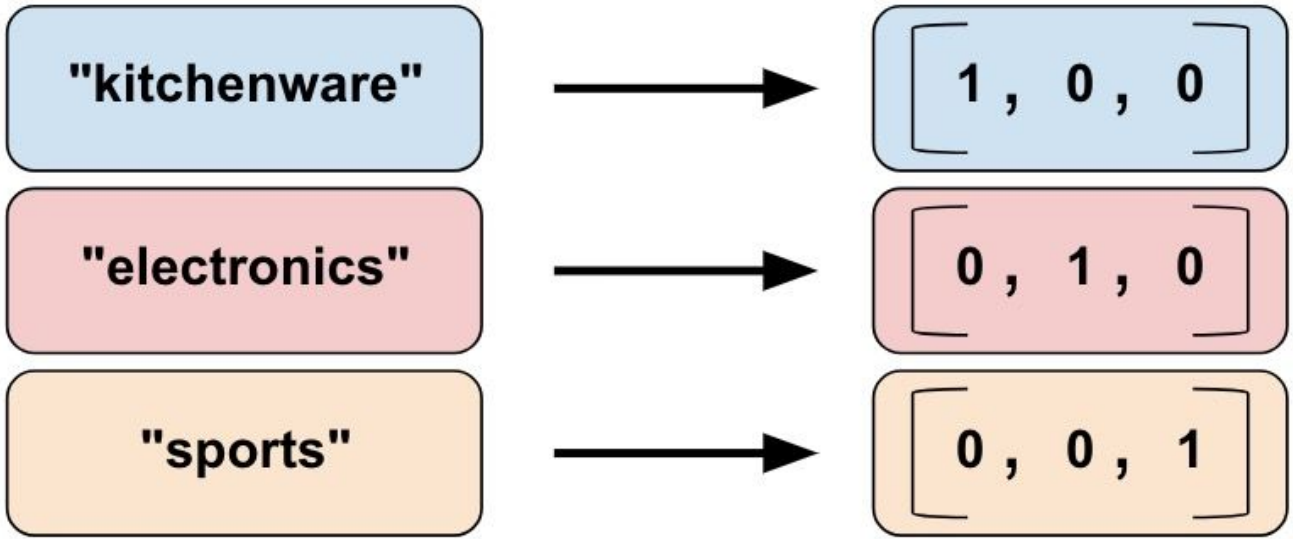
- *one-hot encoding*
- *token embedding* (*word embedding* для слов).



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

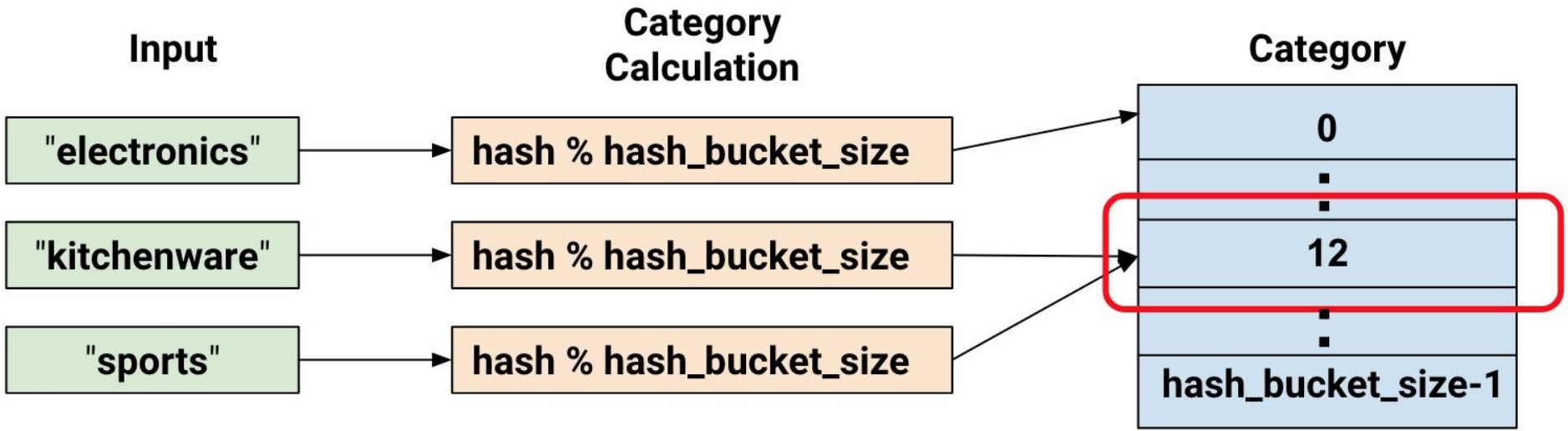
One-hot encoding

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



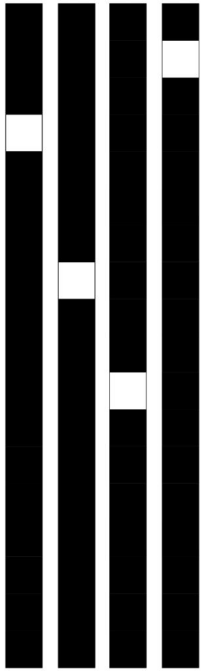
One-hot hashing trick

Удобен при большом словаре

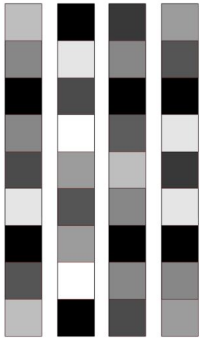


https://www.tensorflow.org/guide/feature_columns

Word embeddings



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

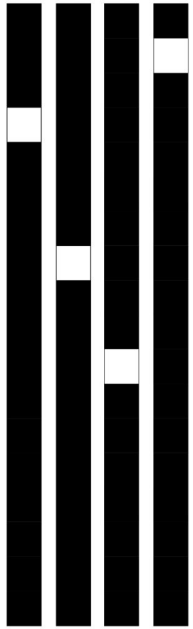


Word embeddings:
- Dense
- Lower-dimensional
- Learned from data

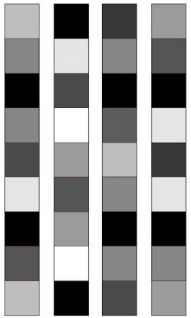
Word embeddings

Наивный вариант: использовать случайные векторы для слов.

Проблема: похожие слова имеют сильно различающиеся вектора.



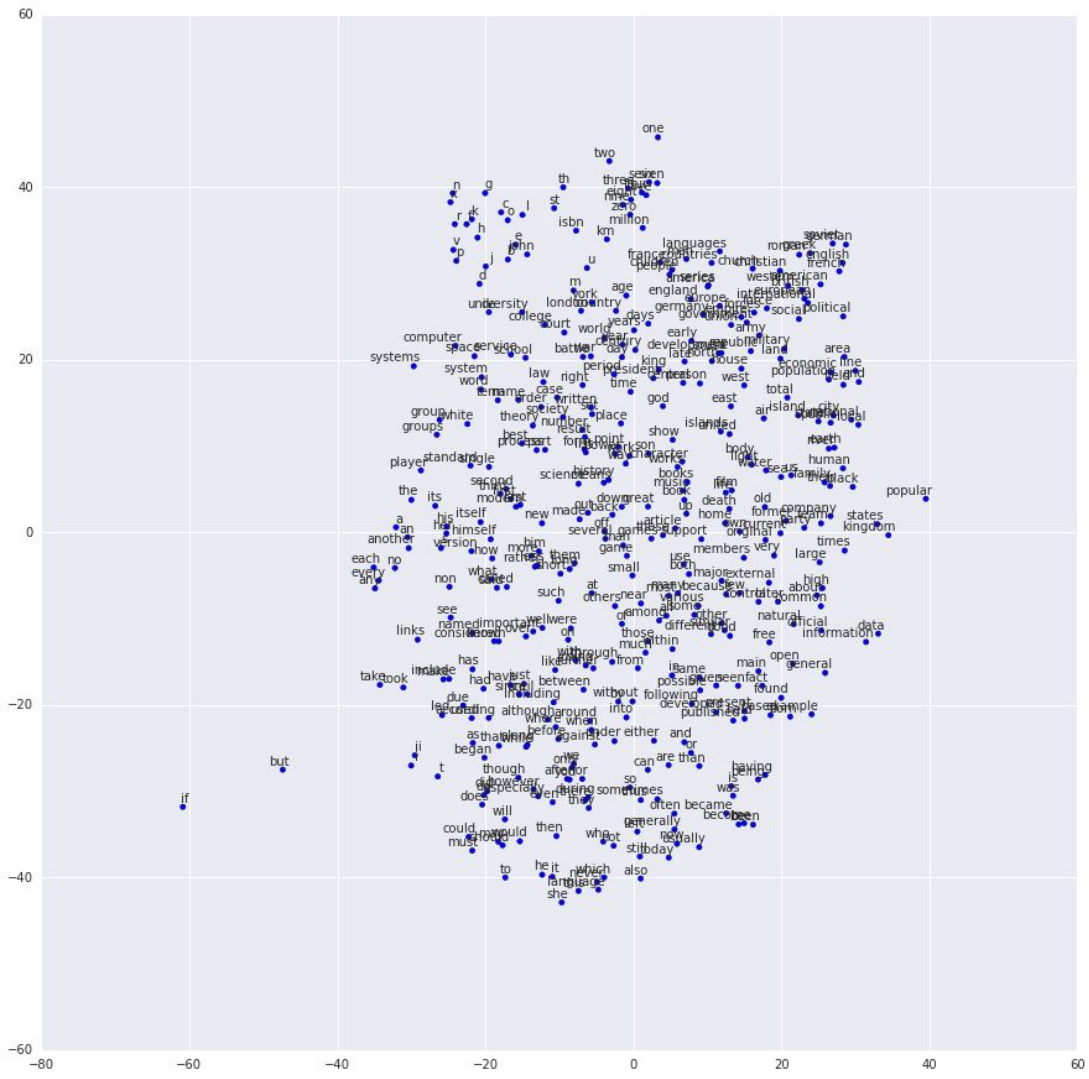
One-hot word vectors:
- Sparse
- High-dimensional
- Hard-coded



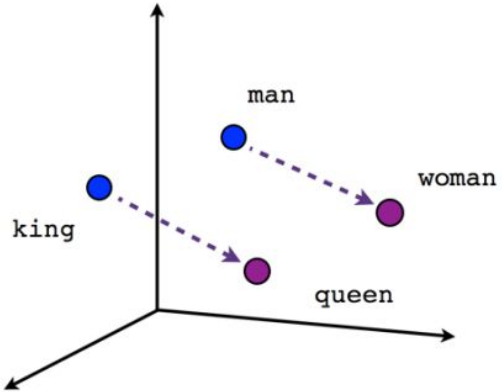
Word embeddings:
- Dense
- Lower-dimensional
- Learned from data

Word embeddings

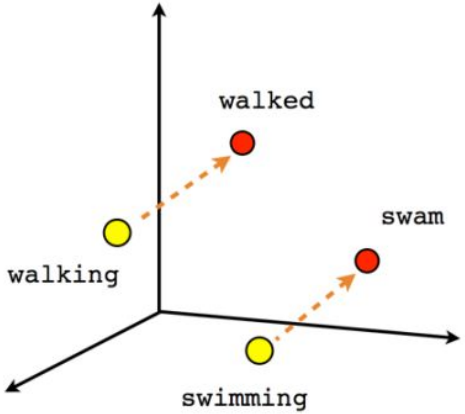
Хотим, чтобы структура этого пространства отражала отношения между словами, а расстояние в этом пространстве соответствовало семантическому расстоянию между словами.



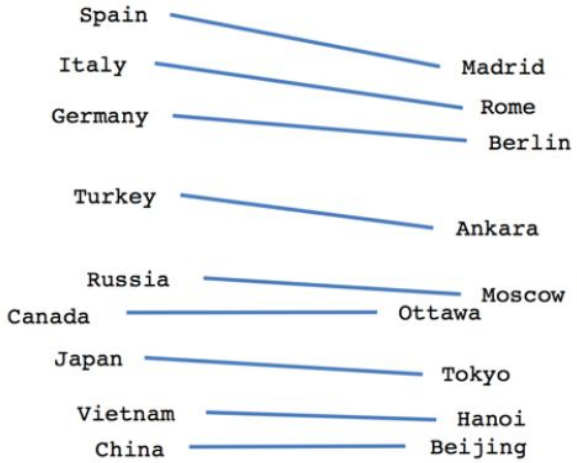
Word embeddings



Male-Female



Verb tense

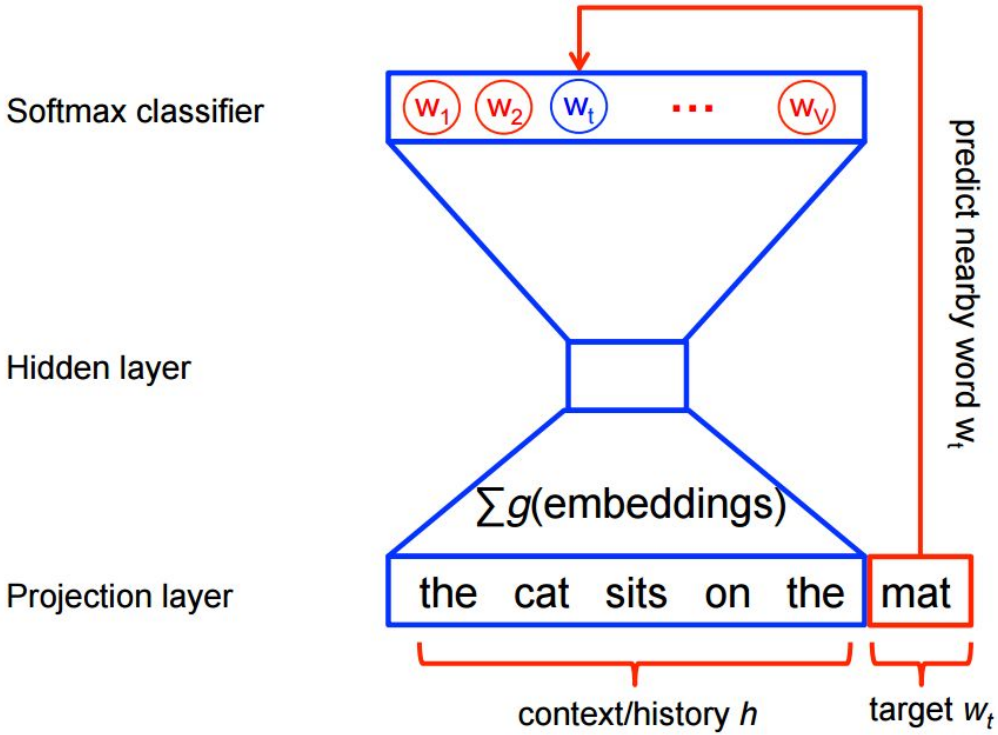


Country-Capital

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

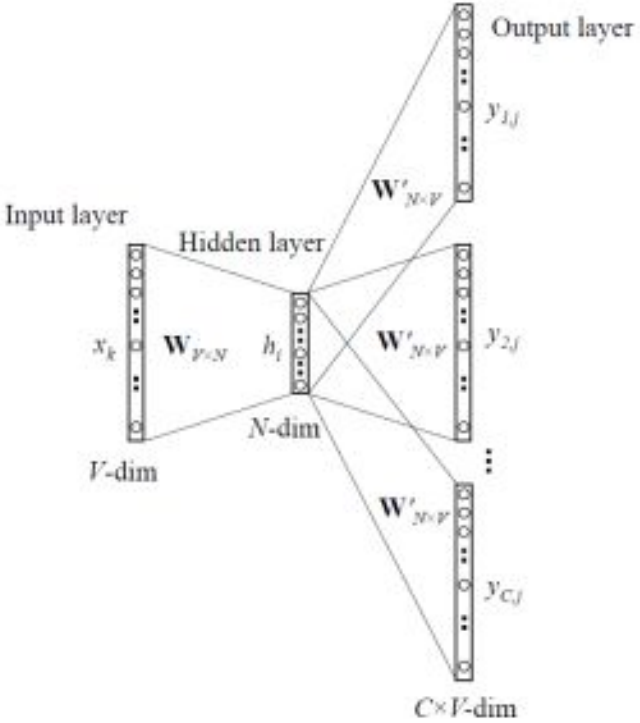
Word embeddings

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

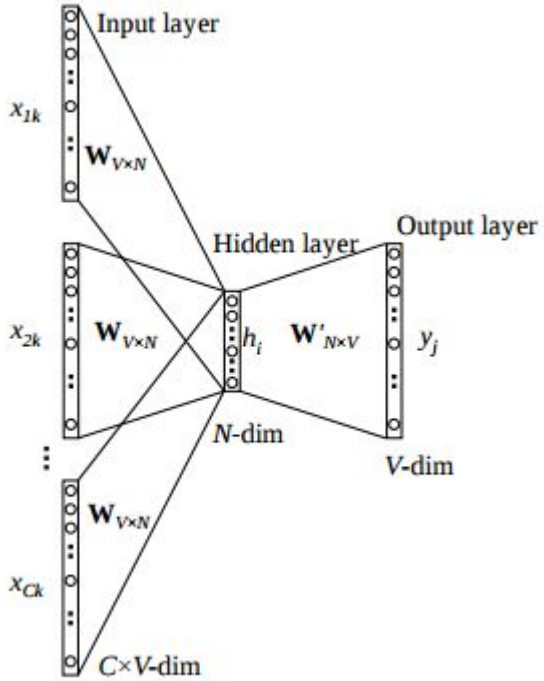


https://www.tensorflow.org/tutorials/representation/word2vec

Word2vec



Skip-gram model



Continuous Bag of Words (CBOW)

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

Word2vec

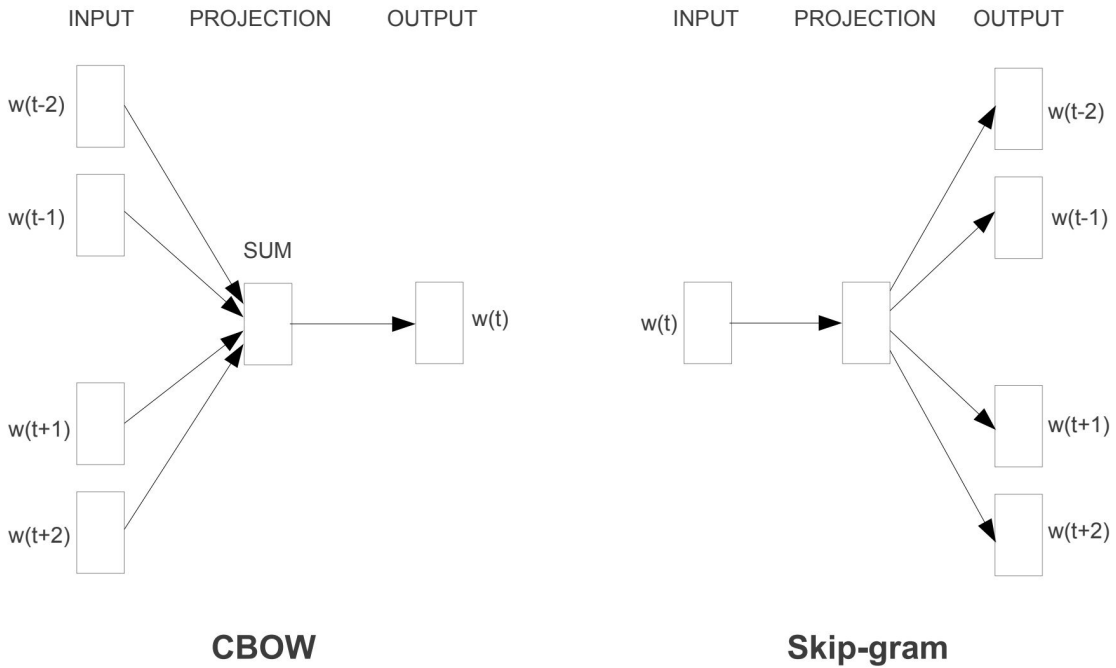


Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

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/
- Использовать пре-тренированные (Word2vec, GloVe, FastText)
  - 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
- 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]]

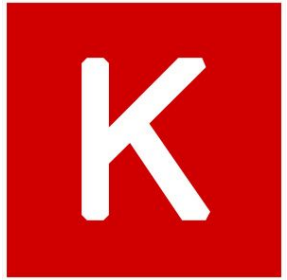
It’s effectively a dictionary lookup:

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/



Практика #1:
Классификация текстов (IMDB)
с помощью RNN

IMDB Movie reviews sentiment classification

https://keras.io/datasets/

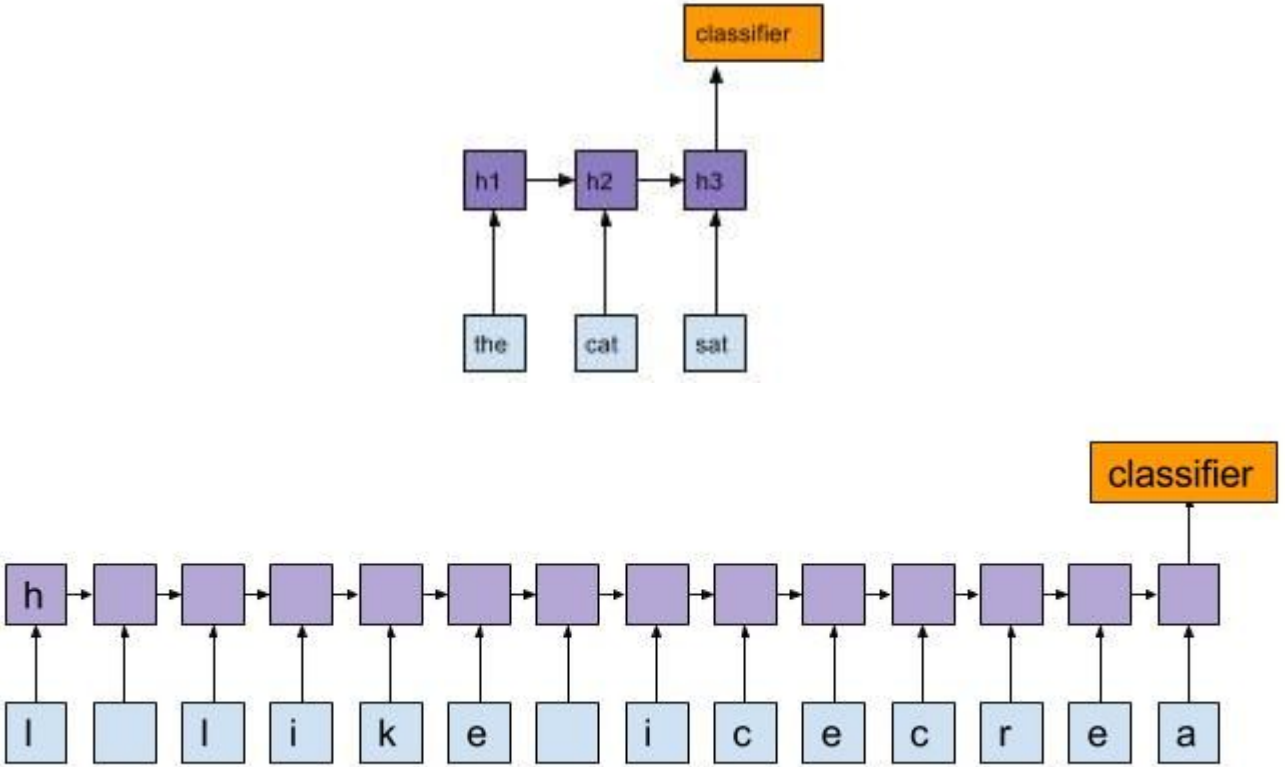
Dataset of 25,000 movies reviews from IMDB, labeled by sentiment (positive/negative). Reviews have been preprocessed, and each review is encoded as a sequence of word indexes (integers). For convenience, words are indexed by overall frequency in the dataset, so that for instance the integer "3" encodes 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.

Ресурсы

- Пример:
  https://raw.githubusercontent.com/fchollet/keras/1.2.2/examples/imdb_lstm.py
  https://raw.githubusercontent.com/fchollet/keras/master/examples/imdb_lstm.py
- Более сложный пример:
  https://github.com/fchollet/keras/blob/1.2.2/examples/imdb_bidirectional_lstm.py
  https://github.com/fchollet/keras/blob/master/examples/imdb_bidirectional_lstm.py
- Разбор похожего кейса
  https://habrahabr.ru/company/dca/blog/274027/

Word-level vs. Character-level



Подготовка данных

```
max_features = 20000
maxlen = 80  # cut texts after this number of words
batch_size = 32


print('Loading data...')
(X_train, y_train), (X_test, y_test) =
imdb.load_data(num_words=max_features)
print(len(X_train), 'train sequences')
print(len(X_test), 'test sequences')
```

Подготовка данных

```
print('Pad sequences (samples x time)')
X_train = sequence.pad_sequences(X_train, maxlen=maxlen)
X_test = sequence.pad_sequences(X_test, maxlen=maxlen)
print('X_train shape:', X_train.shape)
print('X_test shape:', X_test.shape)
```

Создание модели

```
model = Sequential()
model.add(Embedding(max_features, 128))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

Создание модели (GRU)

```
model = Sequential()
model.add(Embedding(max_features, 128))
model.add(GRU(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

Создание модели (Bidirectional)

```
model = Sequential()
model.add(Embedding(max_features, 128))
model.add(Bidirectional(LSTM(128, dropout=0.2,
recurrent_dropout=0.2)))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

Обучение модели

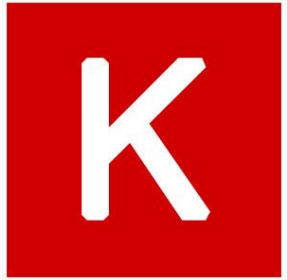
```
print('Train...')
model.fit(X_train, y_train, batch_size=batch_size, epochs=15,
          validation_data=(X_test, y_test))
score, acc = model.evaluate(X_test, y_test,
                            batch_size=batch_size)
print('Test score:', score)
print('Test accuracy:', acc)
```

Практика #2:
Классификация текстов (Twitter)
с помощью RNN

Sentiment analysis на данных Twitter

Задание:

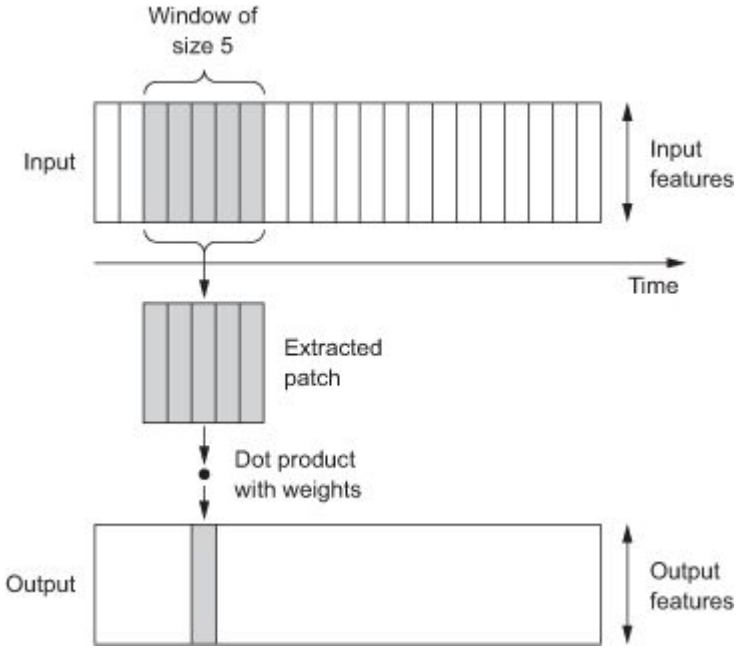
- Взять датасет с позитивными и негативными твитами:
  - http://study.mokoron.com/
- Подготовить датасет для использования нейросетью.
- Сформировать и обучить модель.
- Оценить качество на тестовой выборке.



Практика #3:
Классификация текстов (IMDB)
с помощью CNN

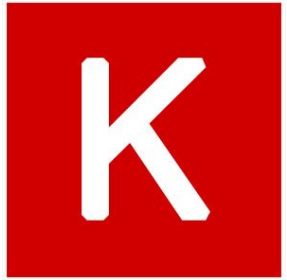
CNN for sequence classification

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



1D CNN example

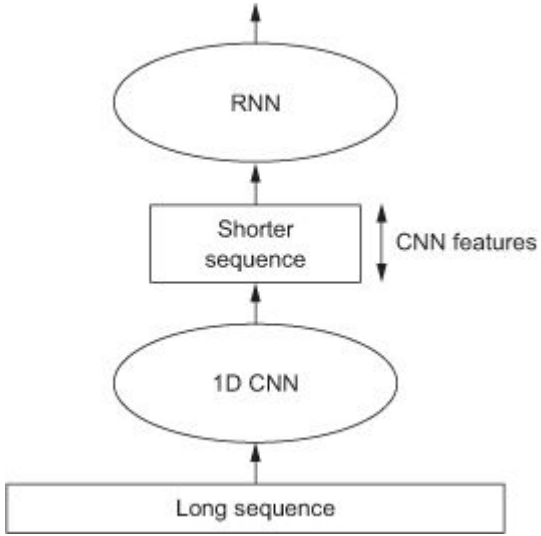
```
from keras.models import Sequential
from keras import layers
from keras.optimizers import RMSprop

model = Sequential()
model.add(layers.Embedding(max_features, 128, input_length=max_len))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.MaxPooling1D(5))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(1))

model.summary()

model.compile(optimizer=RMSprop(lr=1e-4),
              loss='binary_crossentropy',
              metrics=['acc'])
history = model.fit(x_train, y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```

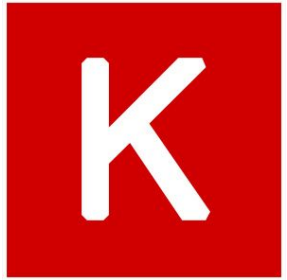


Практика #4:
Классификация текстов (IMDB)
с помощью CNN+RNN

RNN + CNN

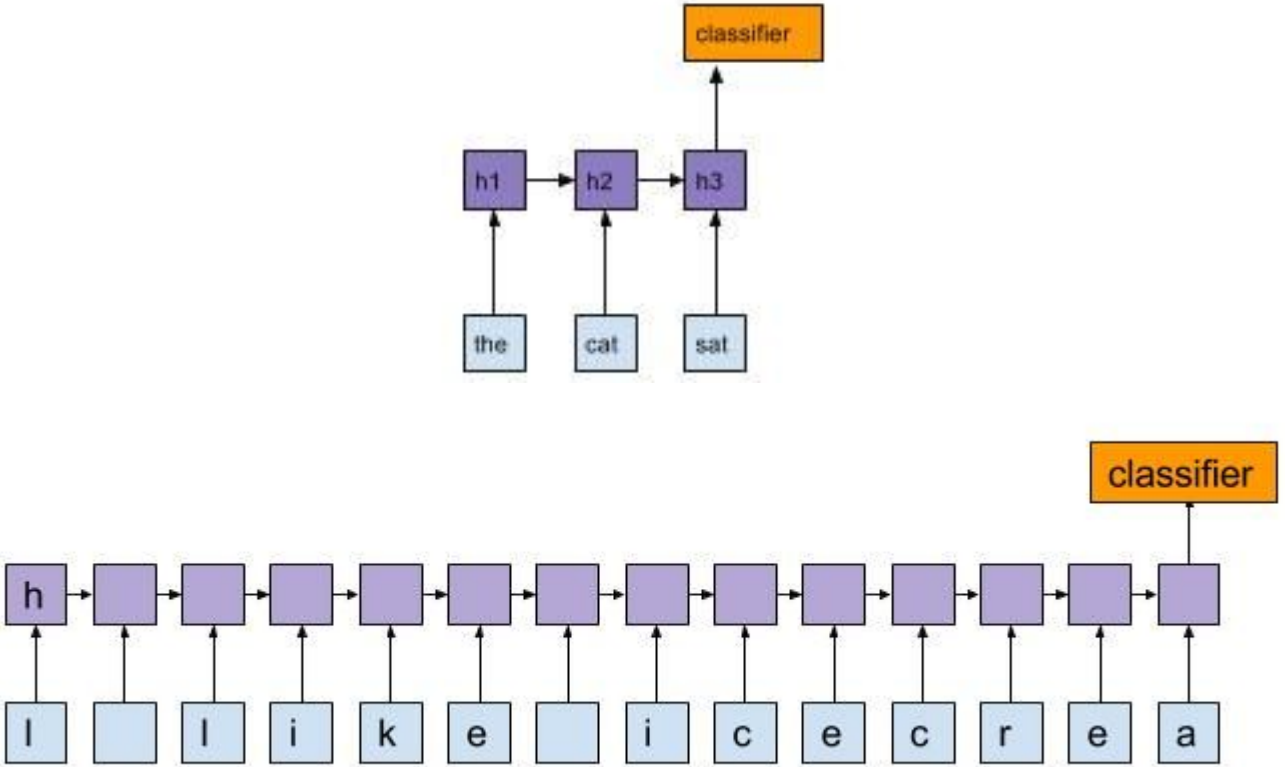
```
from keras.models import Sequential
from keras import layers
from keras.optimizers import RMSprop

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))

model.summary()

model.compile(optimizer=RMSprop(), loss='mae')
history = model.fit_generator(train_gen,
                              steps_per_epoch=500,
                              epochs=20,
                              validation_data=val_gen,
                              validation_steps=val_steps)
```





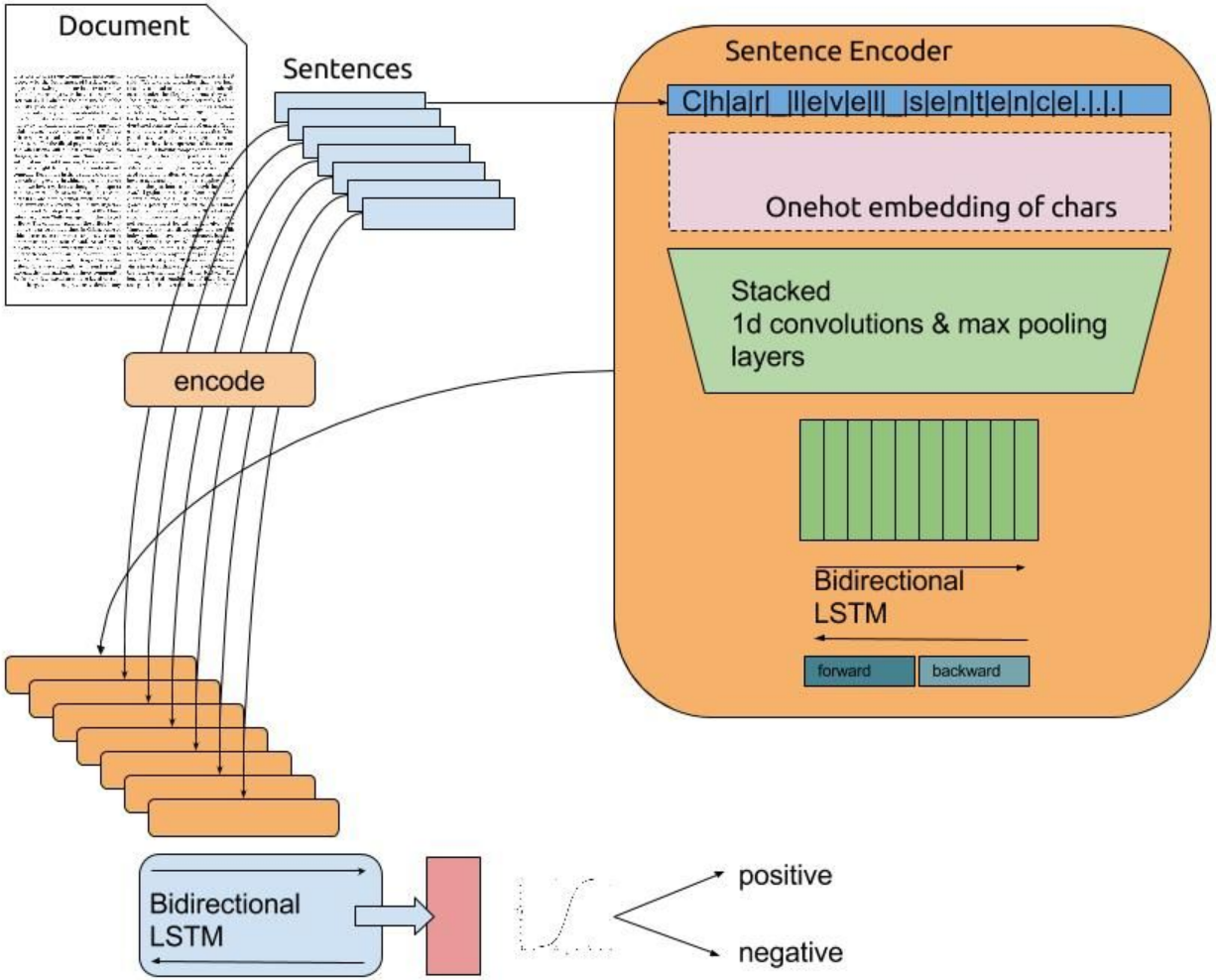
Практика #5:
Классификация текстов (IMDB)
с помощью посимвольной нейросети

Word-level vs. Character-level



Ресурсы

- Датасет с Kaggle https://www.kaggle.com/c/word2vec-nlp-tutorial/data
  https://www.kaggle.com/c/word2vec-nlp-tutorial/download/labeledTrainData.tsv.zip
- How to read: Character level deep learning
  https://offbit.github.io/how-to-read/
  https://github.com/offbit/char-models
- Код для CNN-RNN посимвольной модели:
  https://raw.githubusercontent.com/offbit/char-models/master/doc-rnn2.py
  mkdir checkpoints
- Код для CNN посимвольной модели:
  https://github.com/offbit/char-models/blob/master/doc-cnn4.py
- Ещё про embeddings
  https://towardsdatascience.com/beyond-word-embeddings-part-2-word-vectors-nlp-modeling-from-bow-to-bert-4ebd4711d0ec


Document
Sentences
Sentence Encoder
C|h|a|r|_|l|e|v|e|l|_|s|e|n|t|e|n|c|e|.|.|.|
Onehot embedding of chars
Stacked
1d convolutions & max pooling
layers
encode
Bidirectional
LSTM
forward
backward
Bidirectional
LSTM
positive
negative