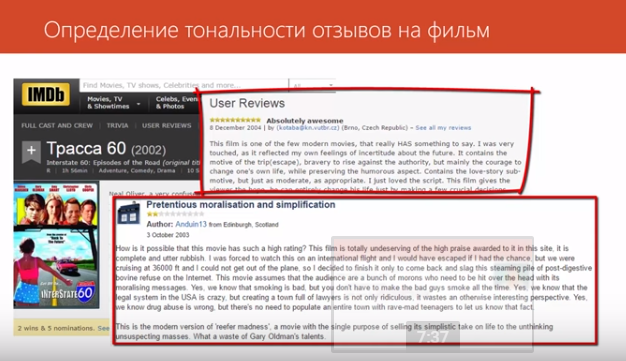
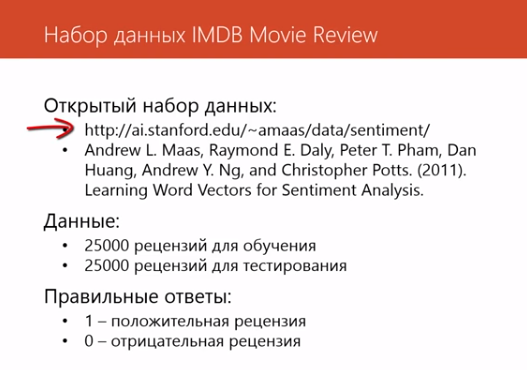
# **10.2. Анализ текстов с помощью рекуррентных нейронных сетей**

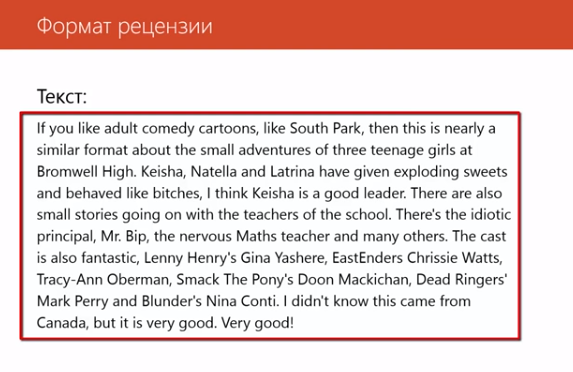
# 

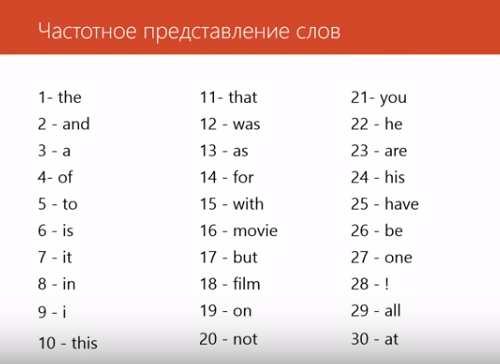
( <https://www.asozykin.ru/courses/nnpython> )

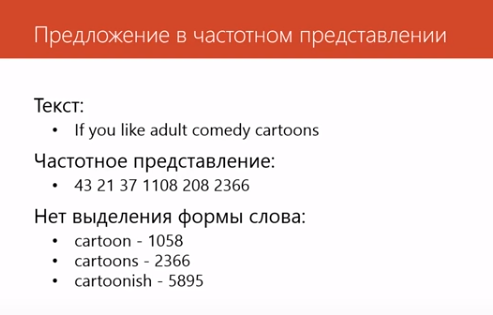
Обучение рекуррентных нейронных сетей в Keras

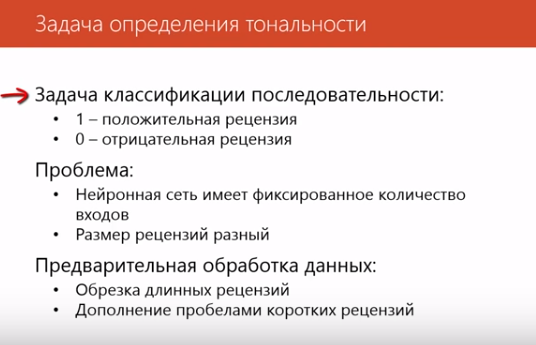


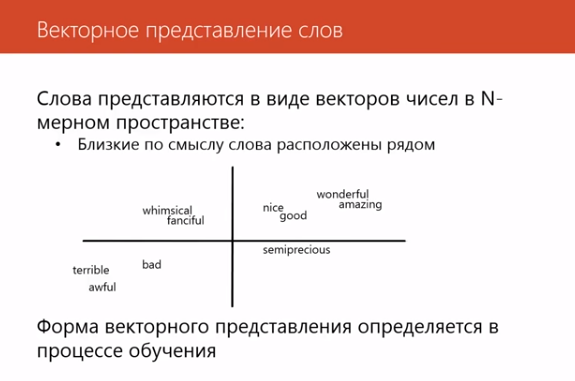


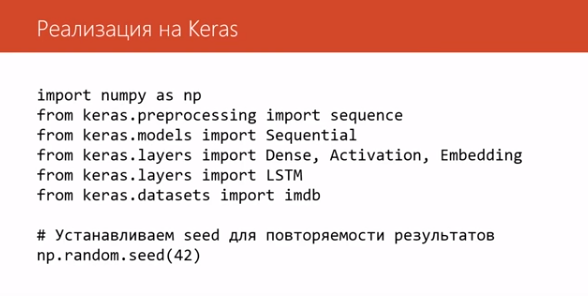


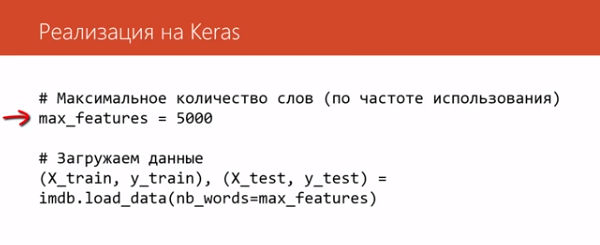


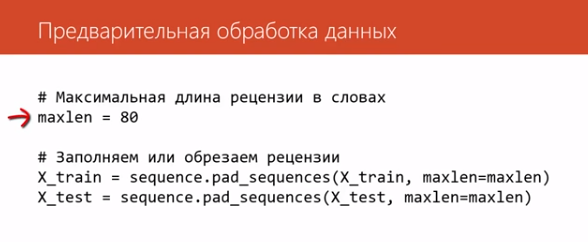


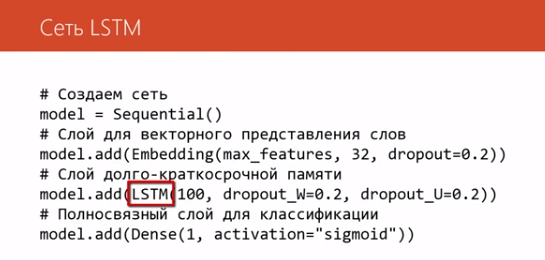


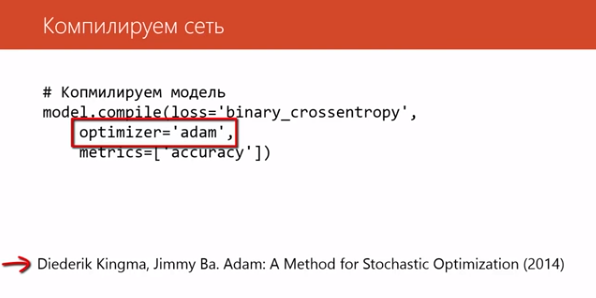


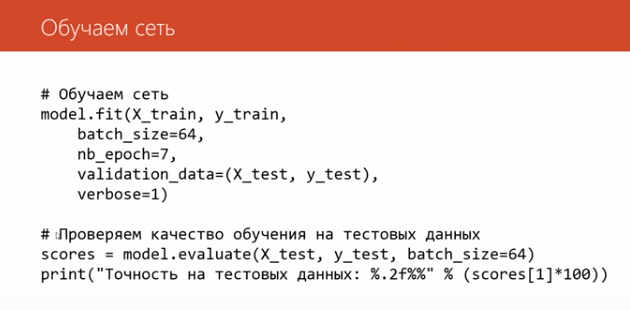




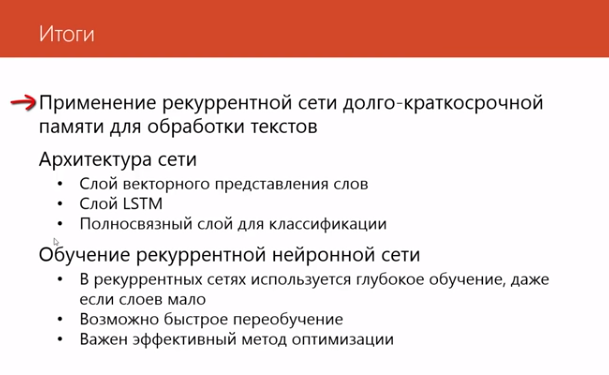












Экспериментируем с гиперпараметрами обучения

**Запишите точность работы сети после обучения на тестовых данных**. Она приводится в последней строке вывода:

Точность работы на тестовых данных: 84.13%

**Проанализируйте точность на проверочной выборке в процессе обучения**. Она указывается после заголовка val\_acc. Началось ли переобучение нейронной сети?

Мы попытаемся улучшить качество обучения сети путем изменения гиперпараметров: количество эпох обучения, количество нейронов на слое LSTM, алгоритм оптимизации.

1. **Количество эпох обучения**. Оценим влияние количества эпох обучения на качество обучения сети. Количество эпох задается в аргументе epochs метода model.fit:
2. model.fit(X\_train, y\_train, batch\_size=64, epochs=7,
3. validation\_data=(X\_test, y\_test), verbose=2)

Попробуйте обучать сеть в течение 5, 10 и 15 эпох. Определите, когда начинается переобучение. Выберите количество эпох, при котором самая высокая точность работы сети на тестовых данных.

1. **Количество нейронов в слое LSTM**. Оцените влияние количества нейронов в LSTM слое сети на качество обучения. Количество нейронов задается при добавлении слоя в модель:
2. model.add(LSTM(XXX, dropout=0.2, recurrent\_dropout=0.2))

Используйте количество нейронов 50, 125, 150. Выберите количество нейронов в LSTM слое, при котором обеспечивается самая высокая точность обучения. Проанализируйте влияния количества нейронов в LSTM слое на время обучения сети.

1. **Алгоритм оптимизации**. Выясним, как влияет алгоритм оптимизации на качество обучения. В программе используется эффективный алгоритм оптимизации [Adam](https://arxiv.org/pdf/1412.6980.pdf). Попробуйте заменить его на стандартный алгоритм стохастического градиентного спуска (Stochastic gradient descent, SGD). Для этого поменяйте значение параметра optimizer при компиляции модели:
2. model.compile(loss='binary\_crossentropy',
3. optimizer='SGD',
4. metrics=['accuracy'])

Выбираем лучшие гиперпараметры

Создайте сеть с лучшими значениями всех гиперпараметров обучения, которые вы определили на предыдущем шаге. Увеличилась ли точность работы сети?

Как сочетание гиперпараметров влияет на время обучения сети? На переобучение?

Что можно сделать, чтобы еще больше увеличить точность?