Методы машинного обучения в задачах анализа текстов

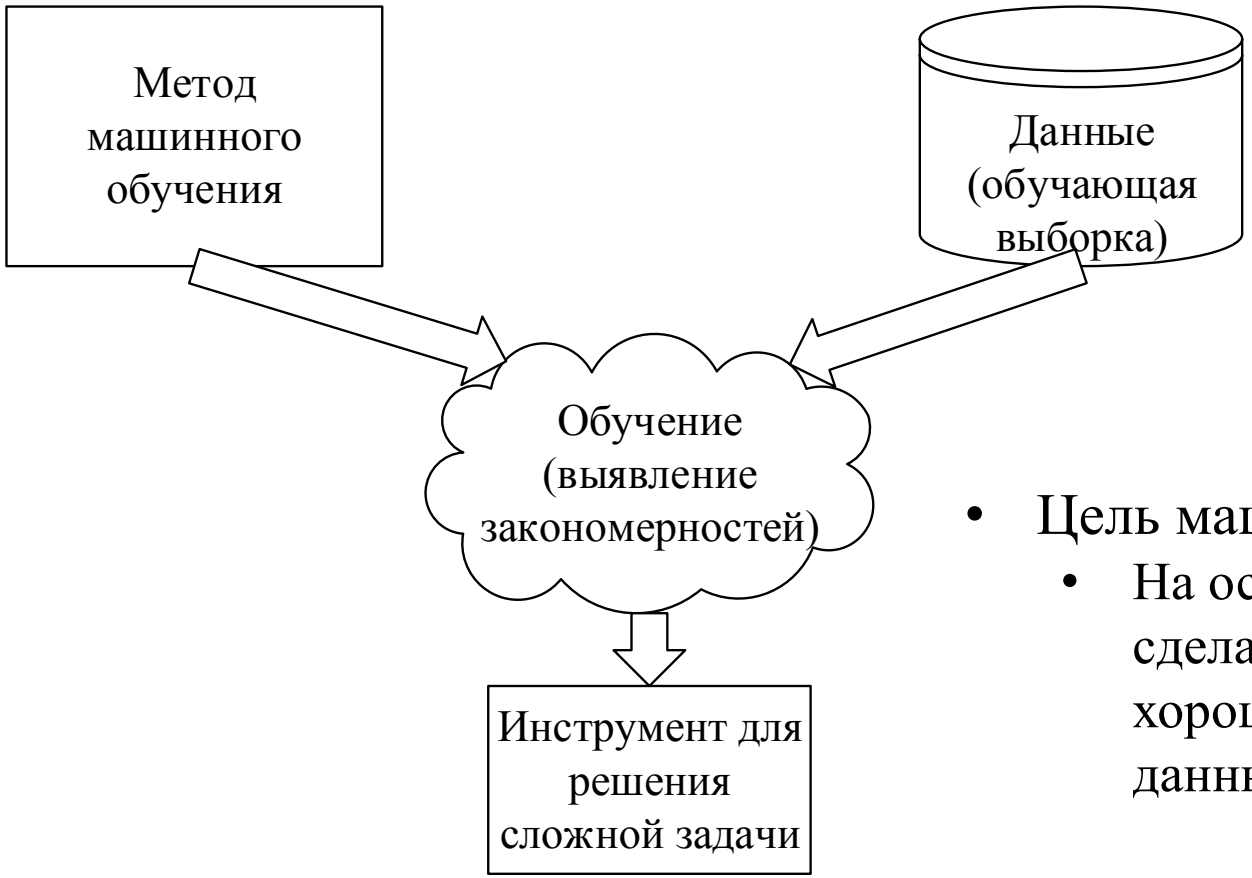
Артем Шелманов, к.т.н.
ИСА ФИЦ ИУ РАН

Москва, 2017

План лекции

- Введение в машинное обучение
- Задача извлечения именованных сущностей (NER)
- Признаки: векторные представления слов (word embeddings)
- Нейронные сети, глубокое обучение, рекуррентные нейронные сети на основе LSTM
- Условные случайные поля (conditional random fields, CRF)
- Модель CRF + LSTM
- Модель CRF + biLSTM + char_embeddings
- Tensorflow
- Live demo (Jupyter)

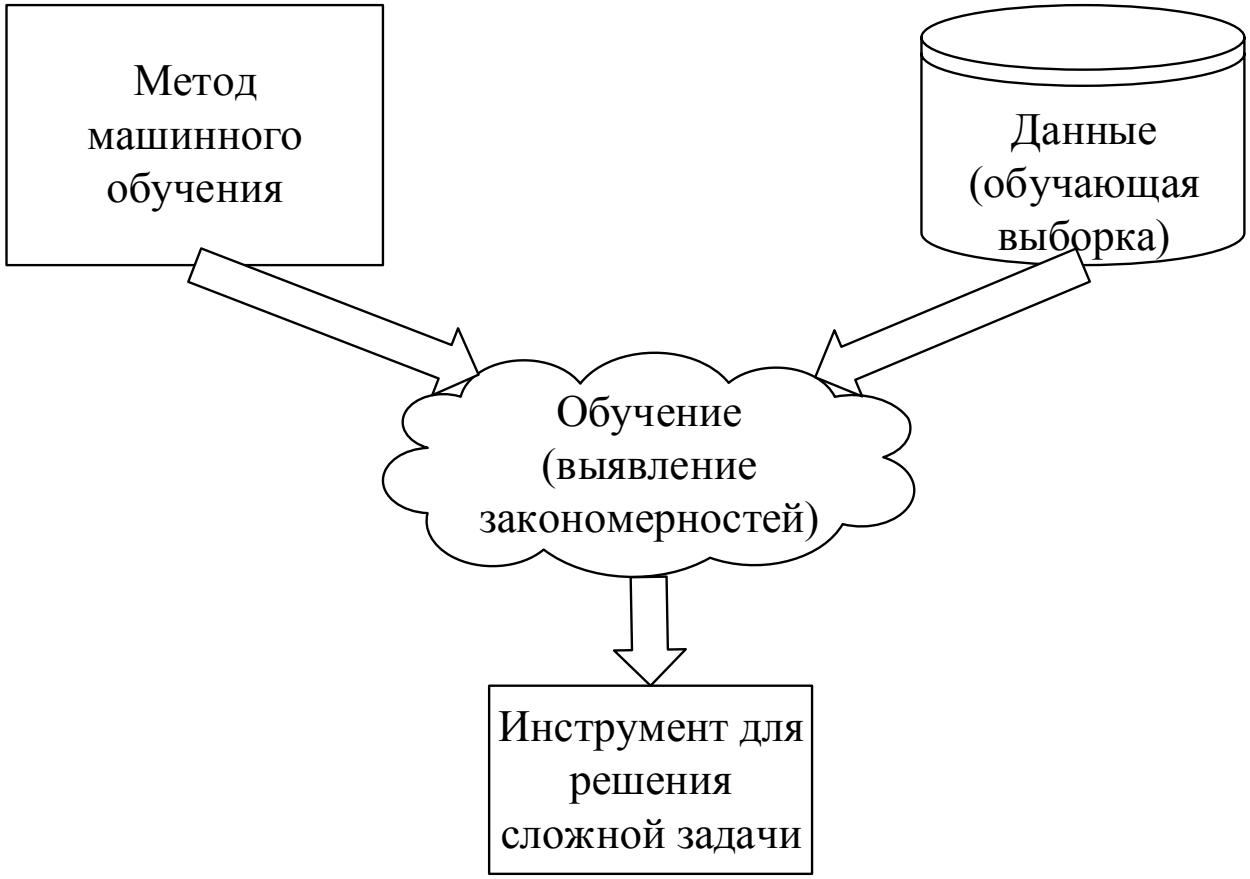
Машинное обучение. Задачи



- Цель машинного обучения:
  - На основе имеющихся данных сделать инструмент, который хорошо решает задачи на новых данных

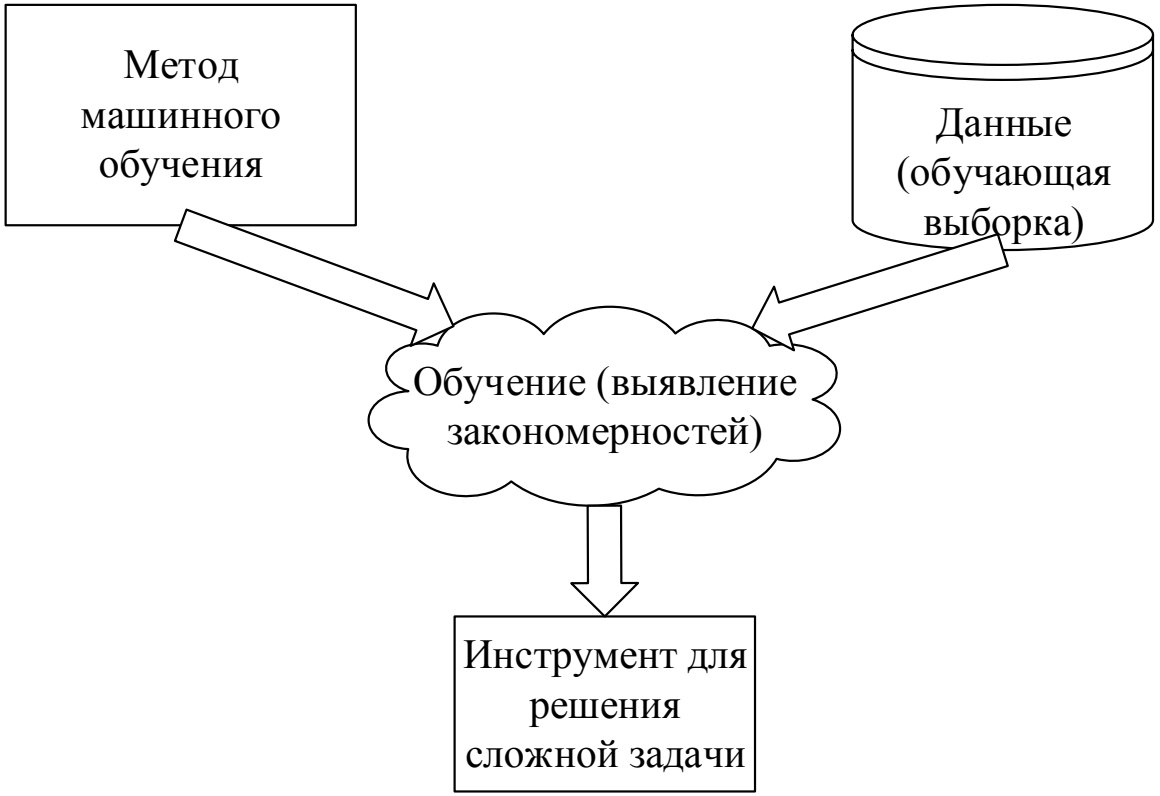
- Машинное обучение нужно, когда:
  - Трудно придумать алгоритм, который решает задачу напрямую
  - Много скрытых закономерностей в данных
  - Закономерности нечеткие (могу проявляться, а могут и нет)

Машинное обучение. Примеры задач



- Примеры:
  - Задача кредитного скоринга в банке: по данным клиента нужно определить давать ли ему кредит (сможет ли он его отдать)
  - Распознавание изображений: по матрице пикселей понять, присутствует ли на изображении объект или нет
  - Извлечение информации из текста: в последовательности символов (или слов) выявить важную для нас информацию (например, географическое название)

Машинное обучение. Обозначения



- X - множество объектов $\{x_1, x_2, \dots x_l\}$
- y - классы (метки) объектов
- (x_i, y_i) - обучающий пример
- l – количество обучающих примеров в выборке
- Совокупность пар $X^l = (x_i, y_i)^l$ - обучающая выборка

Модель машинного обучения $a: X \to Y$, $a(x) = y$

- Каждый объект x_i характеризуется набором признаков $f_j(x_i)$.
- Примеры признаков (задача кредитного скоринга):
  - $f_1(x_i)$ - зарплата клиента
  - $f_2(x_i)$ - количество детей
  - $f_3(\mathrm{x_i})$ - размер непогашенной задолженности

$x_i = (f_1(x_i), f_2(x_i), \dots f_n(x_i))$
$= (100, 1, 1, 0, 0, 0, 0, 1 \ \dots 1)$

Процесс обучения

- Пусть имеется два класса $y \in \{-1,1\}$
- Настраиваем на обучающей выборке линейный классификатор в виде:

$$a(x,w) = sign(\langle w,x\rangle - w_0) = sign\left(\sum_{j=1}^{n} w_j f_j(x) - w_0\right)$$

$w -$ параметры алгоритма

- Отступ (margin) на объекте x_i: $M_i(w) = y_i a(x_i,w)$, если $M_i < 0$ – алгоритм ошибается, $M_i > 0$ – правильный ответ
- Минимизация функционала качества:

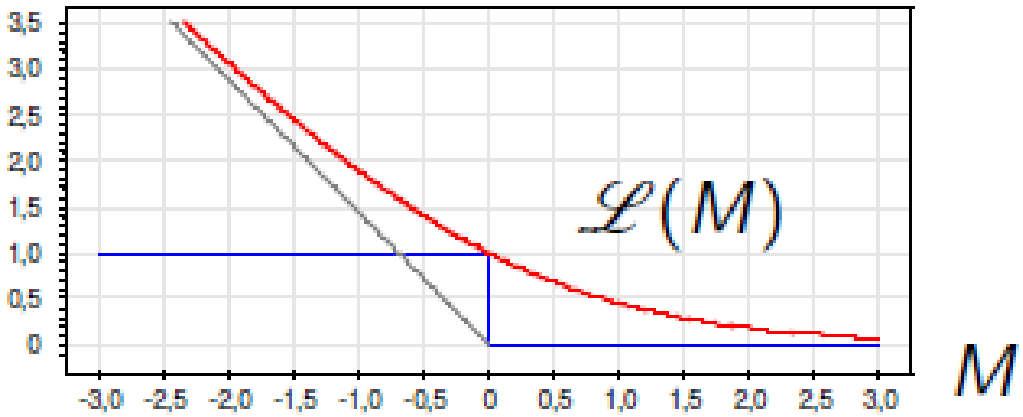
$$[cond] = \begin{cases} 1, cond == true \\ 0, cond == false \end{cases}$$

$$Q\left(w,X^l\right) = \sum_{i=1}^{l} [M_i(w) < 0] \to \min_w$$

Функция потерь

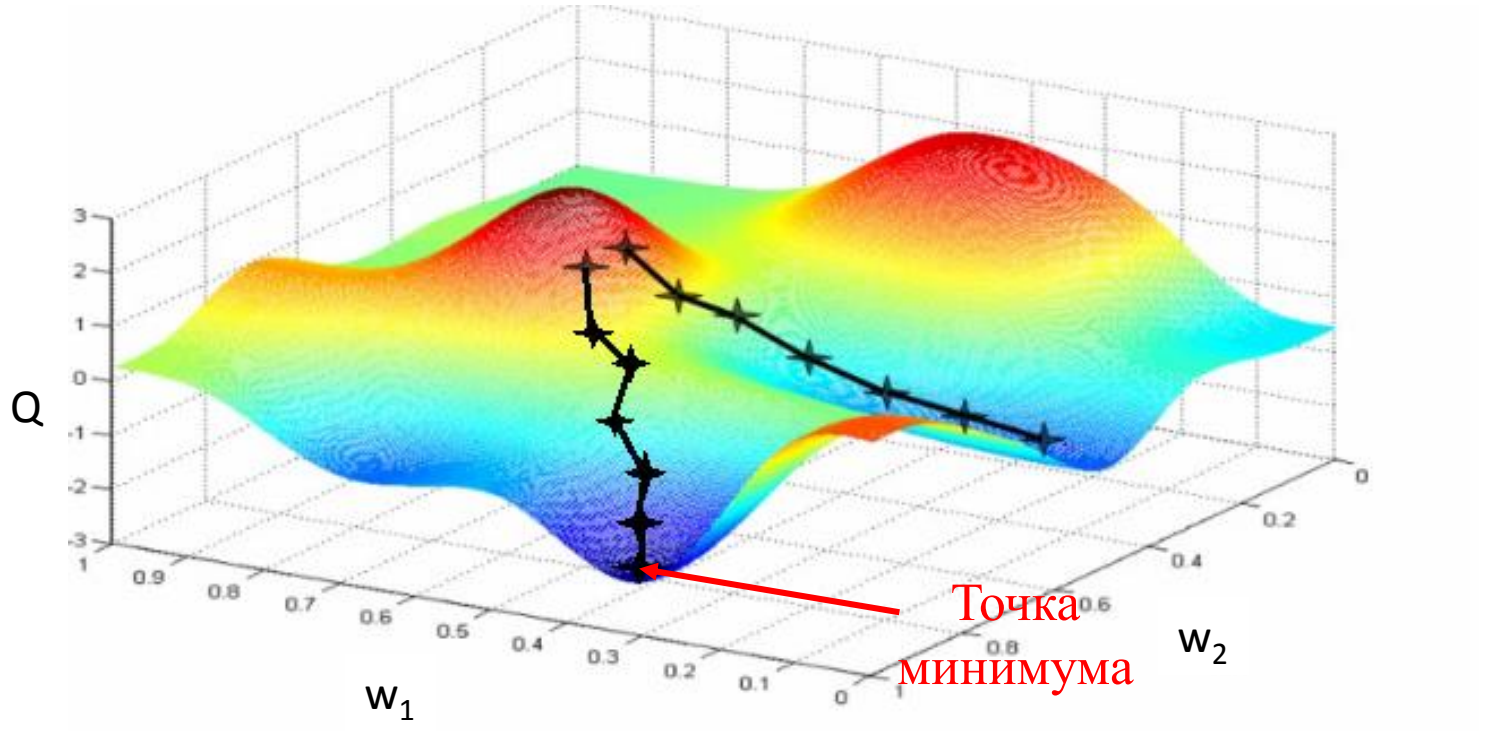
$$\sum_{i=1}^{l} [M_i(w) < 0] \le \tilde{Q}\left(w,X^l\right) = \sum_{i=1}^{l} L\left(M_i(w)\right) \to \min_w$$

Минимизация функционала качества

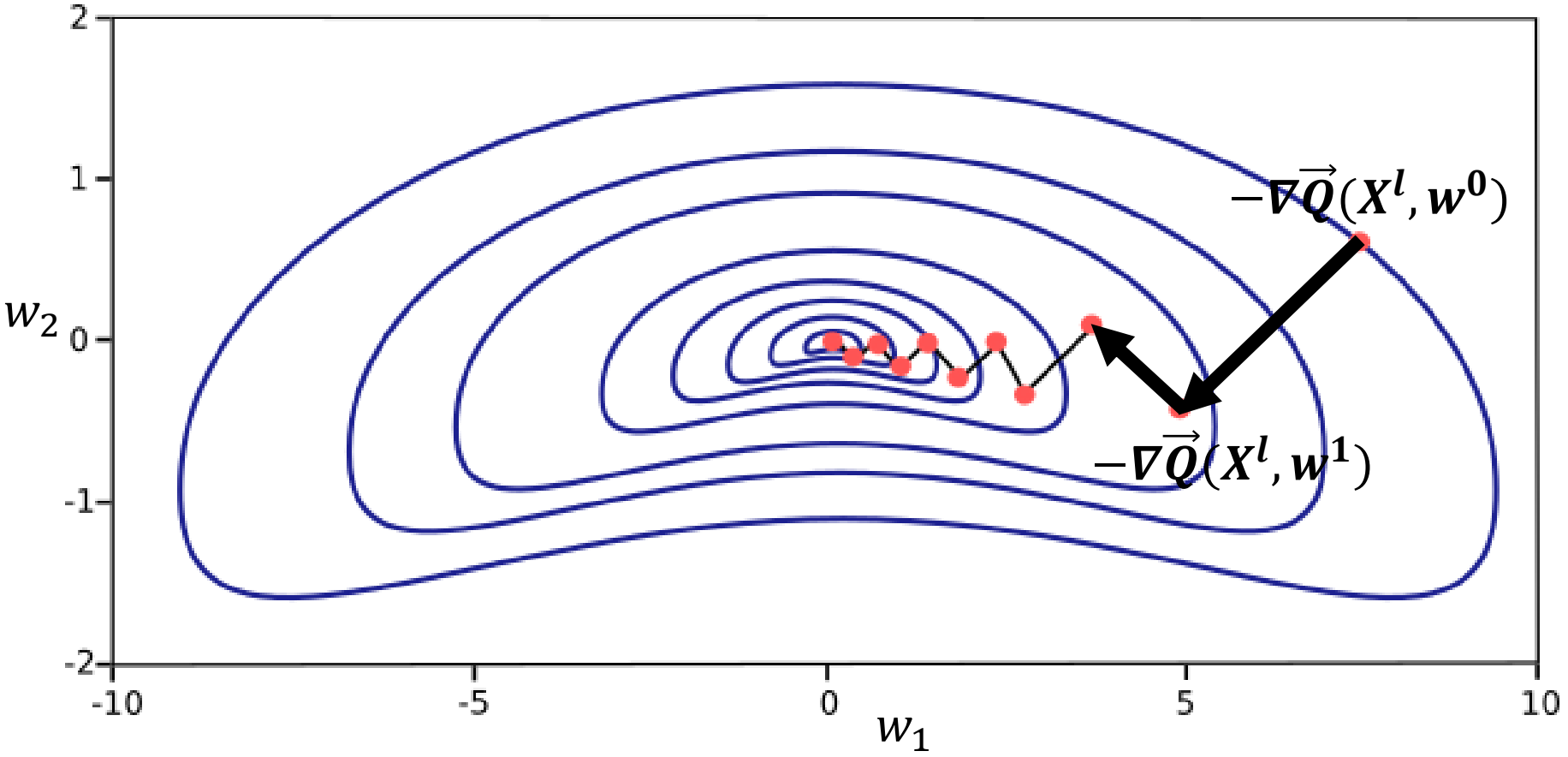


- Функции потерь:
  - $L(M) = \log(1 + e^{-M})$
  - $L(M) = M^2$

- Пример: $\tilde{Q}(w, X^l) = \sum_{i=1}^{l} (y_i a(x_i, w))^2 \to \min_w$



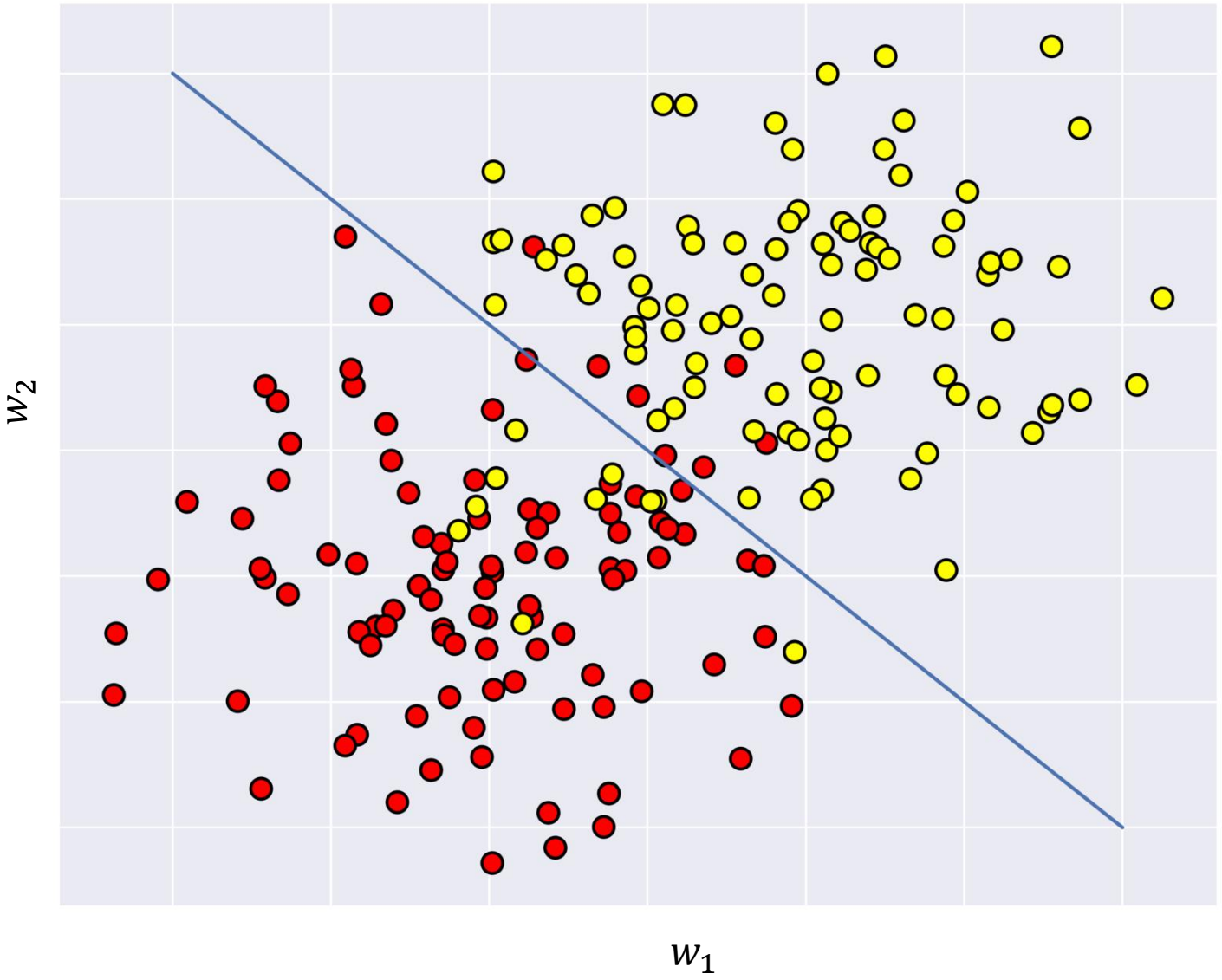
Градиентный спуск



- $\nabla Q(X^l, w) = \left(\frac{\partial Q}{\partial w_0}(x, w), \frac{\partial Q}{\partial w_1}(x, w), \ldots \frac{\partial Q}{\partial w_n}(x, w)\right)$
- $w_{j+1} = w_j - \alpha\nabla Q(X^l, w_j),\ \alpha > 0,\ (\alpha = 0{,}01)$

Разделяющая поверхность

$$a(x,w) = sign(\langle w,x\rangle - w_0) = sign\left(\sum_{j=1}^{n} w_j f_j(x) - w_0\right)$$



Вероятностная постановка задачи машинного обучения

- Восстановить распределение вероятности:
  - $p(x, y)$ - генеративные модели
  - $p(y|x)$ - дискриминативные модели
- Ищем распределение из параметрического семейства $p(x, y) = \varphi(x, y, w)$, параметры w необходимо восстановить по обучающей выборке X^l
- Применяем принцип максимума правдоподобия
  - Ищем такие параметры w, при которых вероятность всей выборки максимальна
  - $p(X^l) = p\left((x^1, y^1), (x^2, y^2), \dots, (x^l, y^l)\right) \overset{\text{iid}}{=} p(x^1, y^1) \dots p(x^l, y^l)$

$$Likelyhood(w, X^l) = \prod_{i=1}^{l} \varphi(x_i, y_i, w) \to \max_w$$

$$-\ln\left(Likelyhood(w, X^l)\right) = -\sum_{i=1}^{l} \ln\left(\varphi(x_i, y_i, w)\right) \to \min_w$$

Методы машинного обучения

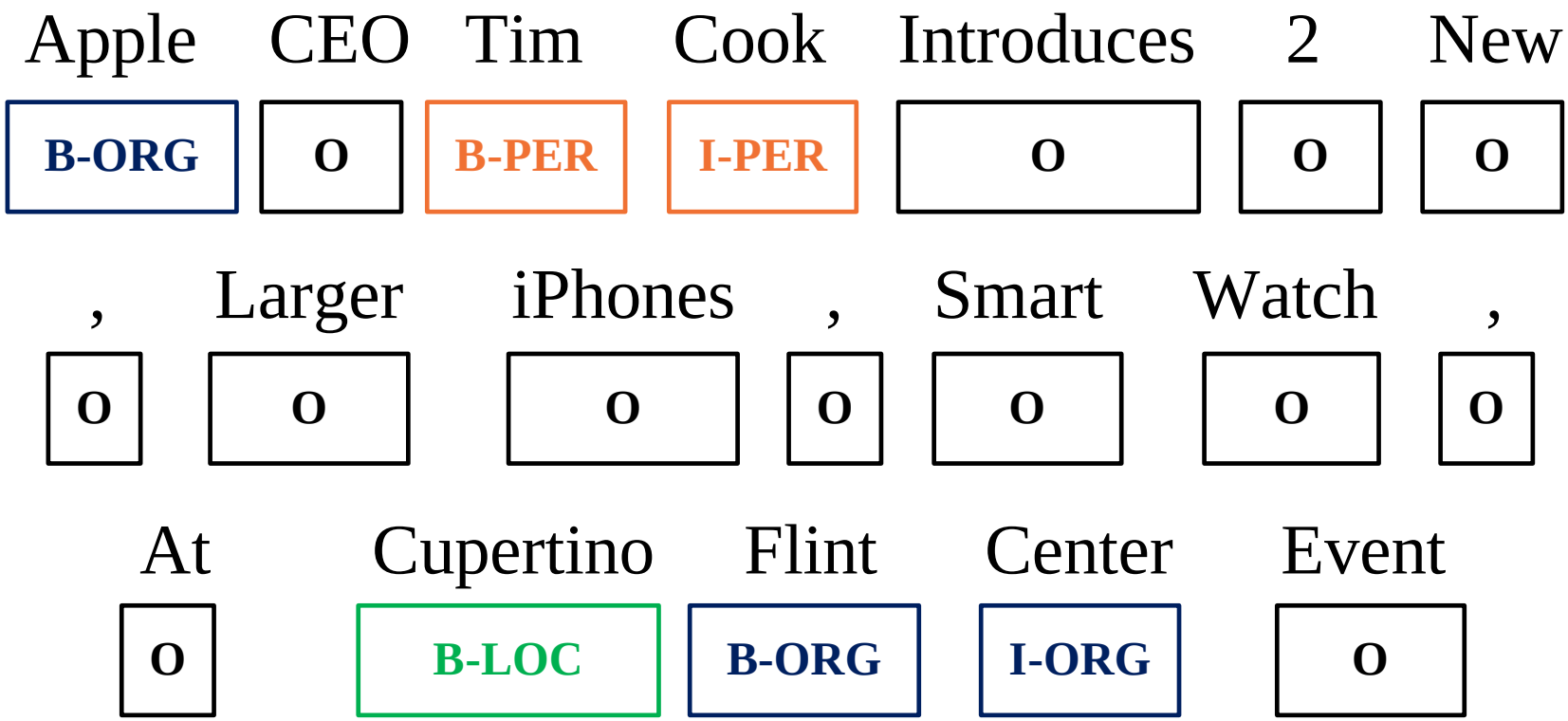
- «Классические»:
  - Линейные методы классификации:
    - Машина опорных векторов SVM
    - Логистическая регрессия
  - Деревья решений
  - Ансамбли:
    - Случайный лес решающих деревьев
    - Градиентный бустинг
  - Наивный байесовский классификатор
  - Модели на основе смеси гауссовских распределений
- Методы обработки последовательностей:
  - Скрытая марковская модель (HMM)
  - Модели на основе условных случайных полей (CRF)
- Нейронные сети (глубокое обучение):
  - Многослойные перцептроны
  - Сверточные сети
  - Рекуррентные нейронные сети

Извлечение именованных сущностей из текстов Named entity recognition (NER)

- Выделить в тексте отрезки, соответствующие объектам с наименованиями
- Понять к каким типам объектов относятся эти отрезки:
  - ФИО персоны
  - Название организации
  - Географическое название

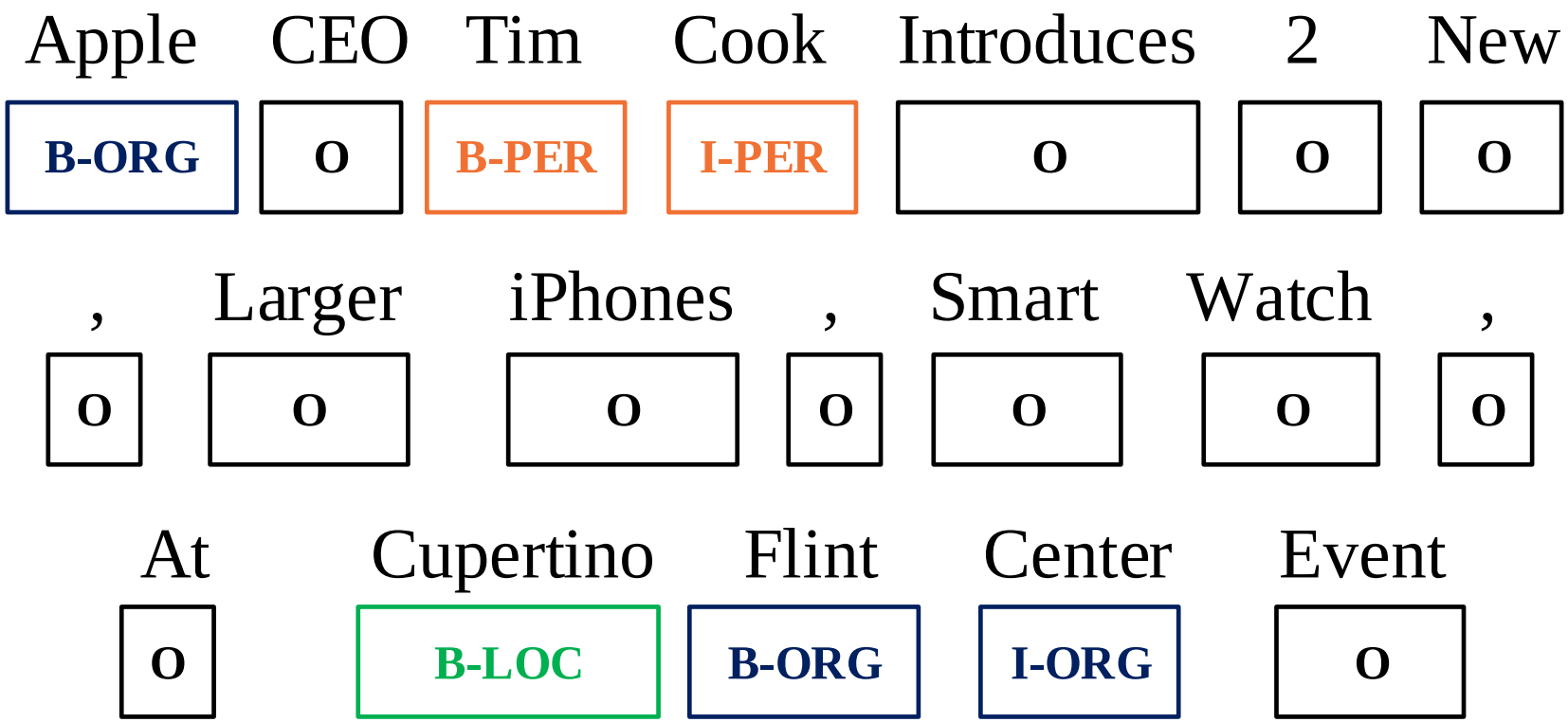
NER как задача машинного обучения

- Разметка BIO: Begin (B-), Inside (I-), Outside (O)
- Пример:
  - B-PER – Токен является началом именованной сущности с типом «персона»
  - I-PER – Токен находится в середине именованной сущности с типом «персона»
  - O – Токен не входит в именованную сущность



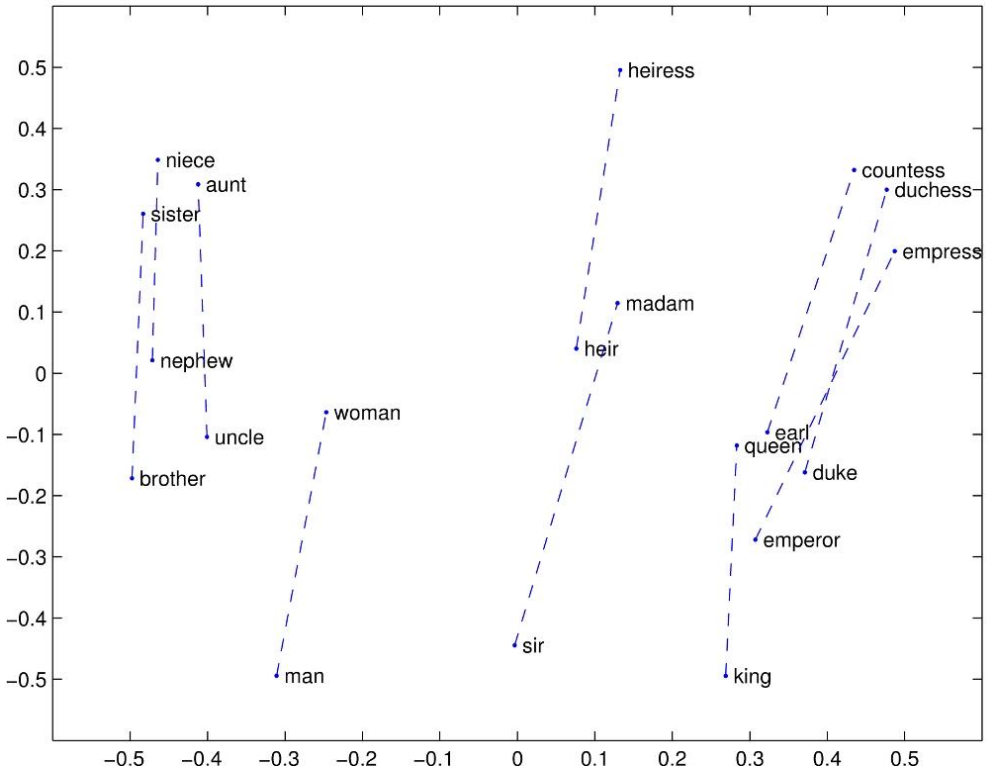
Обучающий корпус

- Корпус соревнования CoNLL-2003
  - https://www.clips.uantwerpen.be/conll2003/ner/
  - https://github.com/synalp/NER/tree/master/corpus/CoNLL-2003
- Разметка BIO



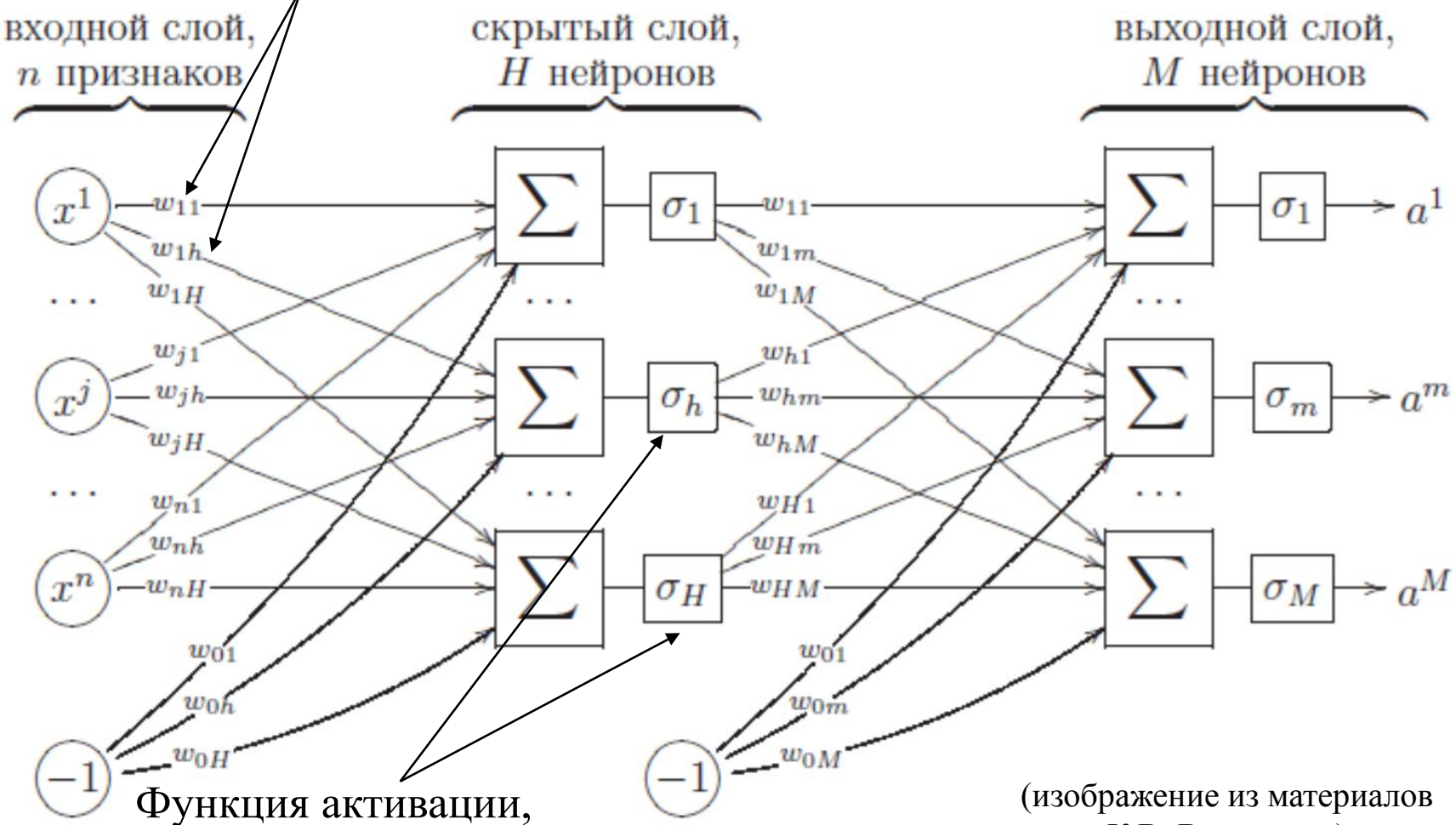
Признаки для обучения: векторные представления слов (word embeddings)

- Векторные представления слов:
  - emb(«банк») = (0,91; 0,25; -0,11; … 0,87) (размерность обычно 50-1000)
- Кодируют смысловую близость слов
- При линейном преобразовании близкие вектора преобразуются в семантически близкие
- Строятся на большим неразмеченным корпусам: 10-100 млрд словоупотреблений
- Инструменты для построения:
  - Word2vec **(Mikolov et al., 2013)**
  - GloVe (Pennington et al., 2014)
- Предобученные модели:
  - Для английского: https://nlp.stanford.edu/projects/glove/
  - Для русского: http://rusvectores.org/ru/models/



(изображение с сайта проекта GloVe)

Нейронные сети (граф математических операций)



Функция активации,
в данном случае сигмоида:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

(изображение из материалов К.В. Воронцова)

Обучение нейронной сети (1)

- Выходные значения сети $a^m(x_i), m = 1..M$ на объекте x_i и промежуточные значения сети $u^h(x_i)$:
  - $a^m(x_i) = \sigma_m\left(\sum_{h=0}^{H} w_{hm} u^h(x_i)\right)$
  - $u^h(x_i) = \sigma_h\left(\sum_{j=0}^{J} w_{jh} f_j(x_i)\right)$
- Эмпирический риск:
  - $Q = \sum_{i=1}^{l} L(a^m(x_i), y_i)$
- По правилу вычисления производной сложной функции:

$$\frac{\partial Q}{\partial w_{hm}} = \sum_{i=1}^{l} \frac{\partial L(a^m(x_i), y_i)}{\partial a^m} \frac{\partial a^m(x_i)}{\partial w_{hm}}$$

$$\frac{\partial Q}{\partial w_{jh}} = \sum_{i=1}^{l} \frac{\partial L(a^m(x_i), y_i)}{\partial u^h} \frac{\partial u^h(x_i)}{\partial w_{hm}}$$

Обучение нейронной сети (2)

- Пусть $L(a(x), y) = \frac{1}{2}\sum_{m=1}^{M}(a^m(x_i) - y_i)^2$, тогда
- Ошибка на выходном слое:

$$\frac{\partial L_i(w)}{\partial a^m} = a^m(x_i) - y_i^m = \varepsilon_i^m$$

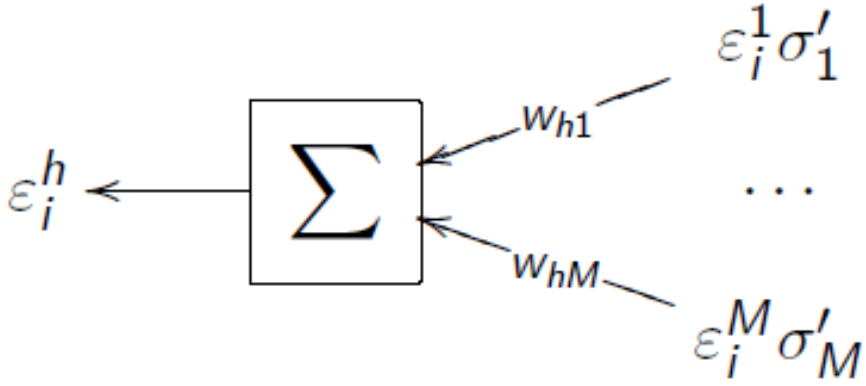
- Ошибка на скрытом слое

$$\frac{\partial L_i(w)}{\partial u^h} = \sum_{m=1}^{M}(a^m(x_i) - y_i^m))\sigma_m' w_{hm} = \sum_{m=1}^{M}\varepsilon_i^m \sigma_m' w_{hm} = \varepsilon_i^h$$

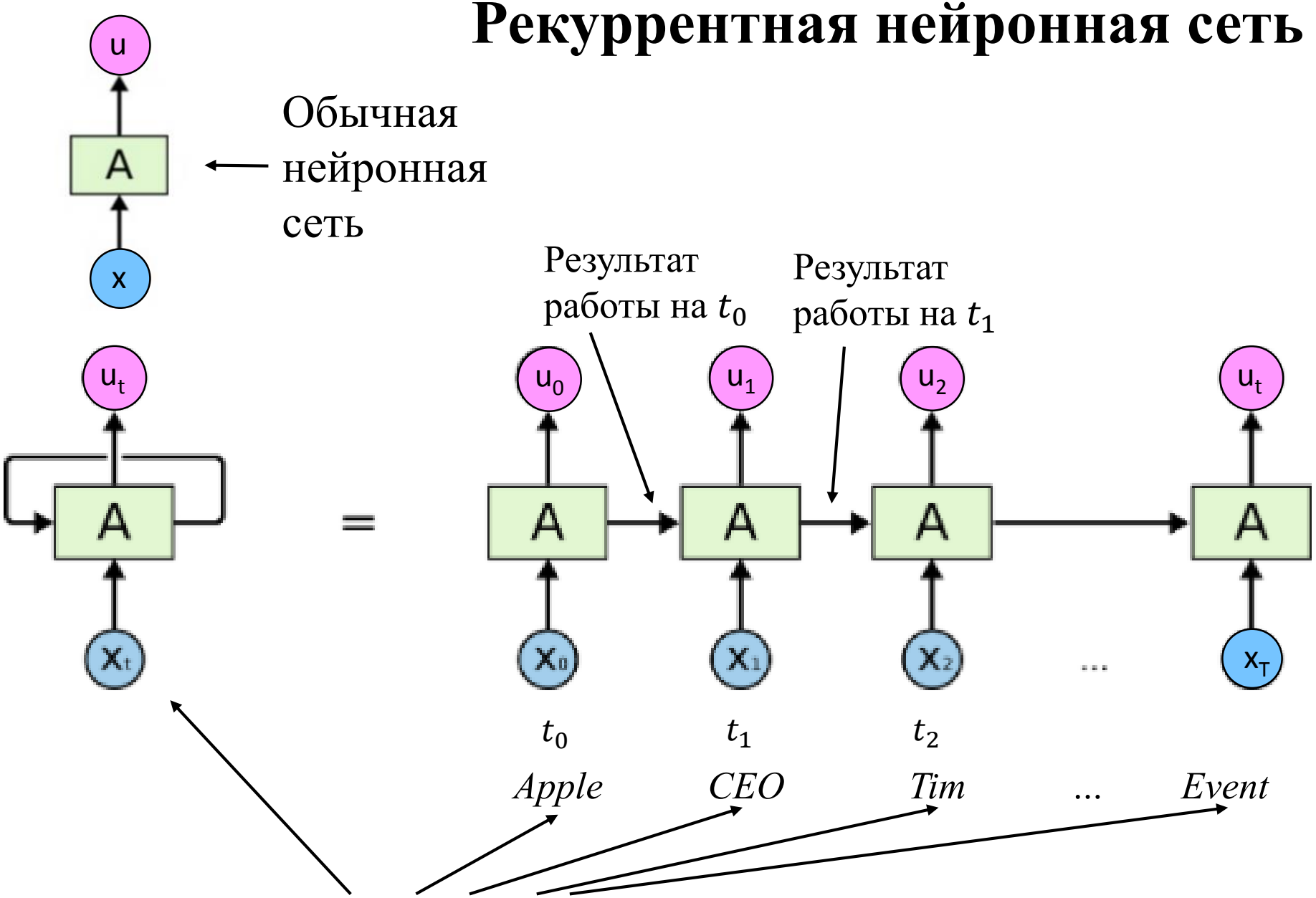
$$\frac{\partial Q}{\partial w_{hm}} = \sum_{i=1}^{l}\varepsilon_i^m\,\sigma_m' u^h(x_i)$$

$$\frac{\partial Q}{\partial w_{jh}} = \sum_{i=1}^{l}\varepsilon_i^h\,\sigma_h' f_j(x_i)$$

Алгоритм обратного распространения ошибки:

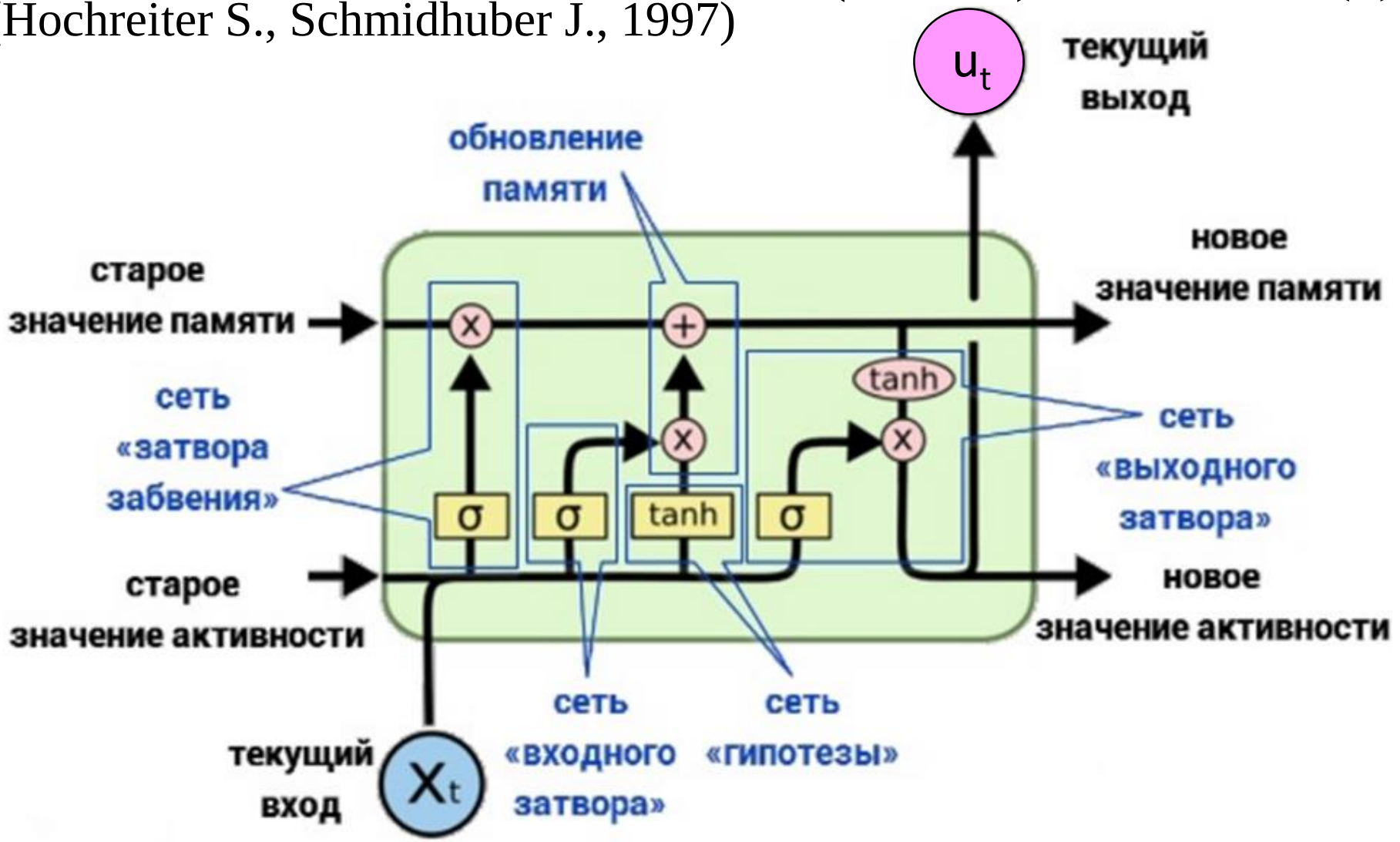


Рекуррентная нейронная сеть



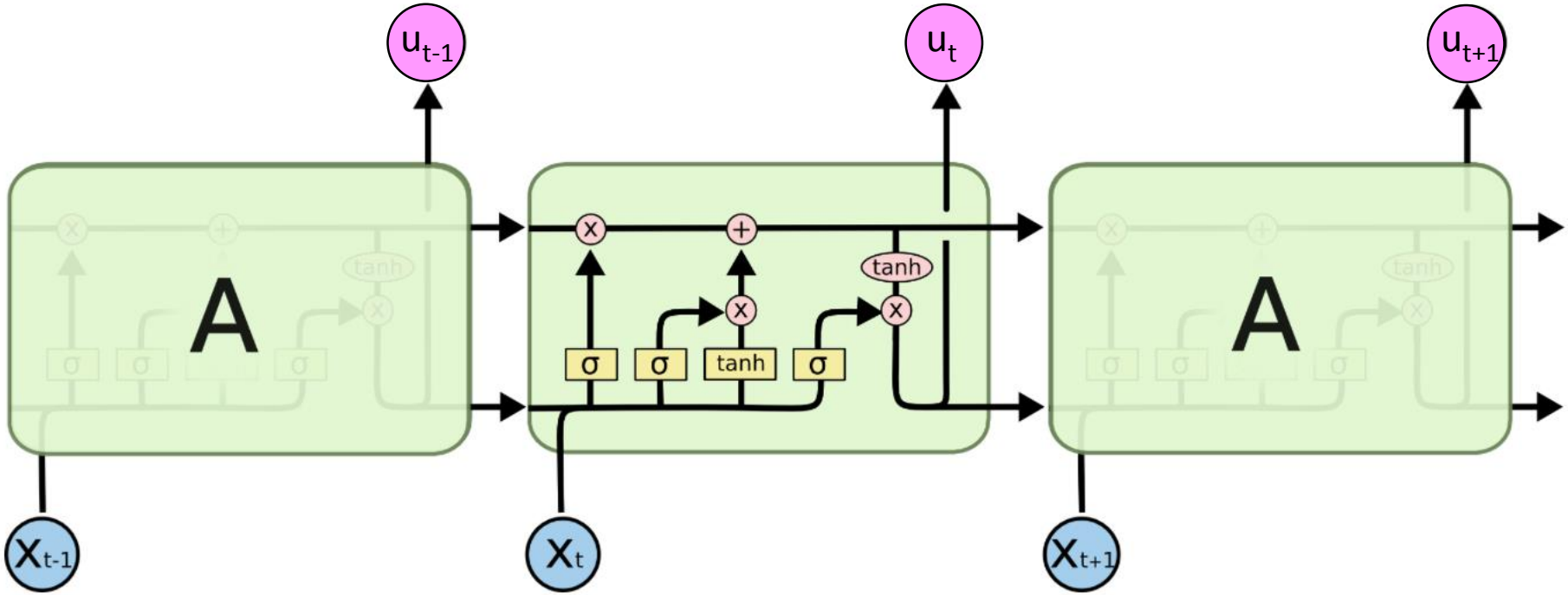
Рекуррентная нейронная сеть с long-short memory (LSTM) ячейками (1)

(Hochreiter S., Schmidhuber J., 1997)



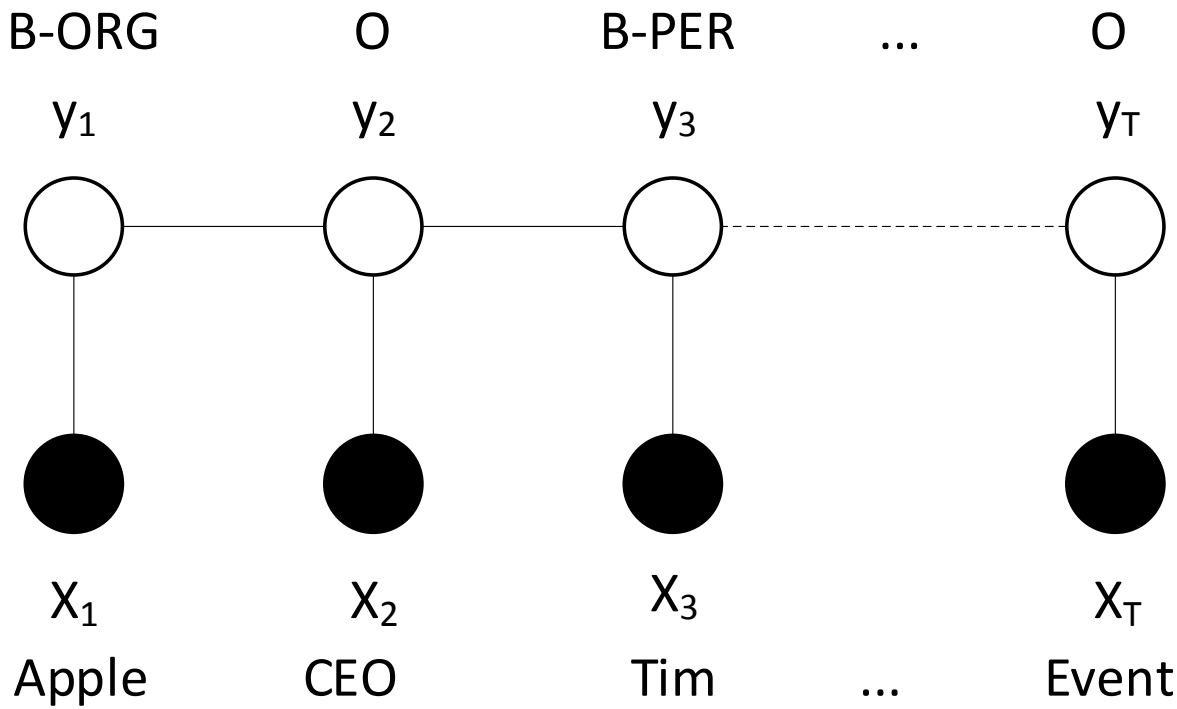
- LSTM решает проблему gradient vanishing

Рекуррентная нейронная сеть с LSTM ячейками (2)



Линейные условно случайные поля (Linear-chain CRF)

- Графическая вероятностная модель линейного условно случайного поля: (Lafferty J., McCallum A. et al., 2001)



- CRF позволяют учитывать метки (классы) соседних слов и глобально максимизировать вероятность всей последовательности меток

Линейные условно случайные поля

- Вычисление вероятности:
  - Для вычисления применяем динамическое программирование

$$p(y|x) = \frac{1}{Z(x)} \prod_{t=1}^{T} \exp\left(\sum_{k=1}^{n} w_k f_k(y_t, y_{t-1}, x_t) \right)$$

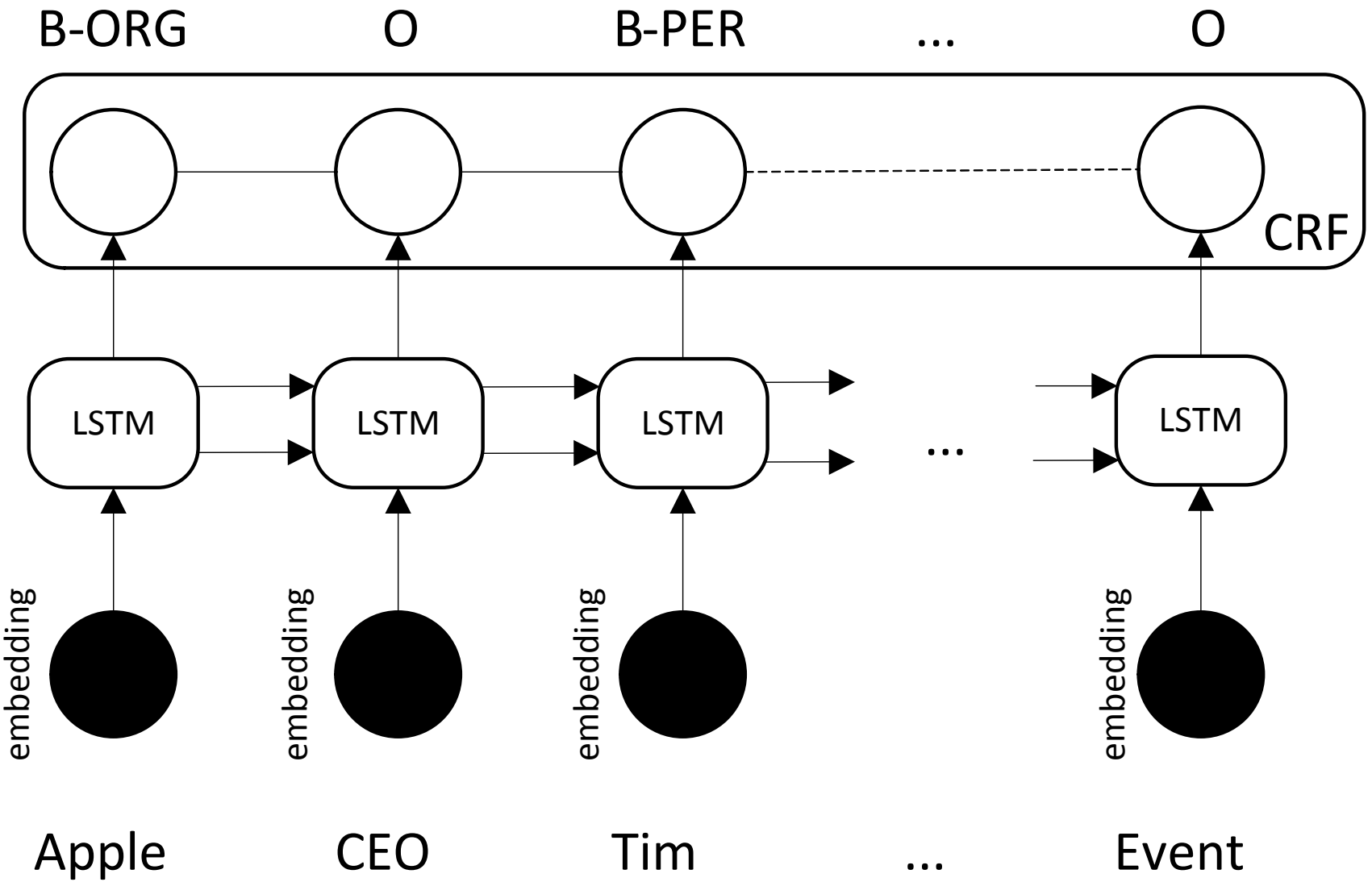
$$Z(x) = \sum_{y} \prod_{t=1}^{T} \exp\left(\sum_{k=1}^{n} w_k f_k(y_t, y_{t-1}, x_t) \right)$$

- Обучение:
  - Максимизация правдоподобия с помощью градиентного метода:

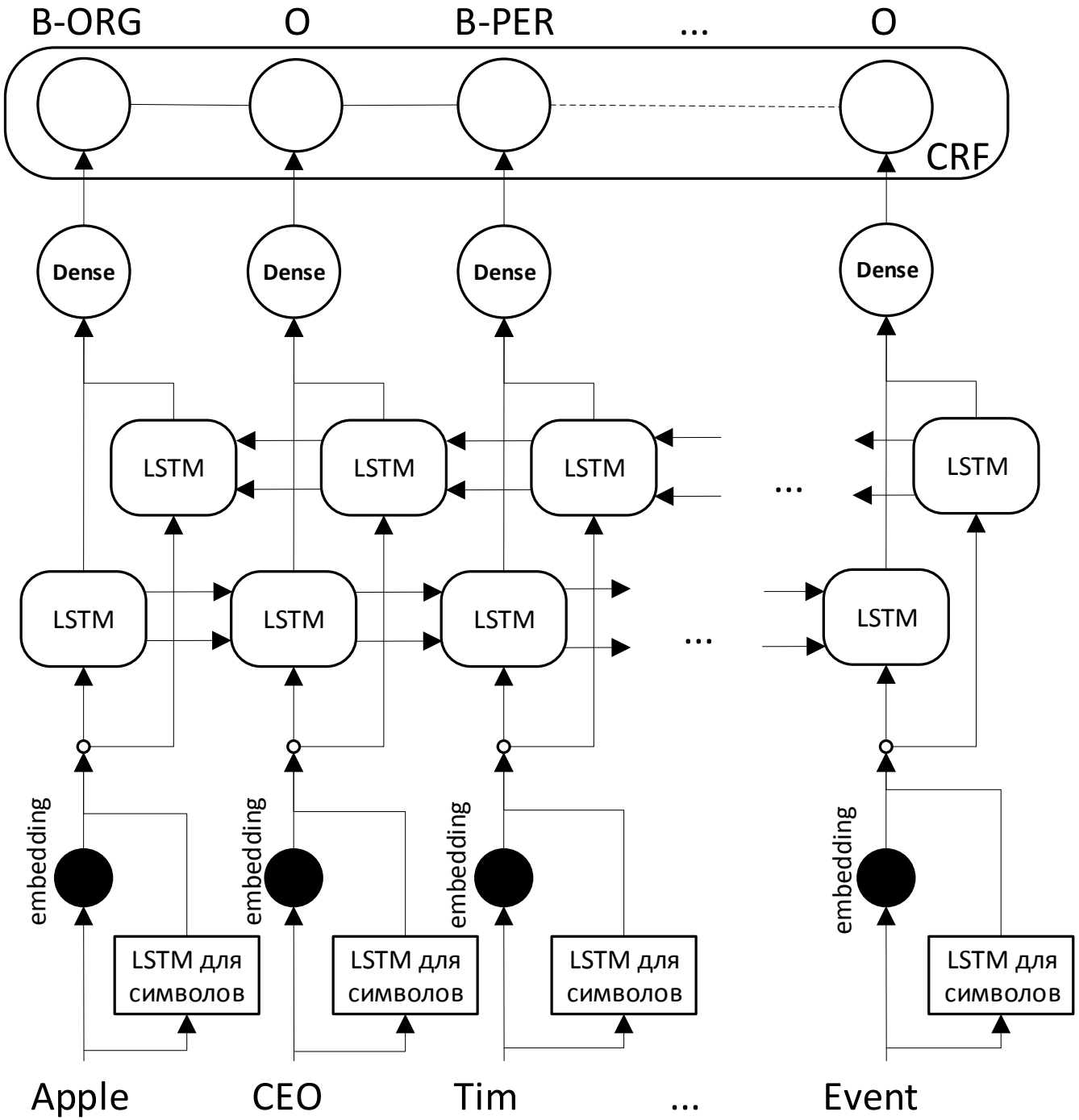
$$CRF\_log\_likelyhood\left(X^l, w\right) = \sum_{i=1}^{l} \sum_{t=1}^{T} \sum_{k=t}^{n} w_k f_k(y_t^i, y_{t-1}^i, x_t^i) - \sum_{i=1}^{l} \log\left(Z\left(x^i\right)\right)$$

$$CRF\_log\_likelyhood\left(X^l, w\right) \rightarrow \max_{w}$$

Модель, совмещающая рекуррентные нейронные сети и случайные поля

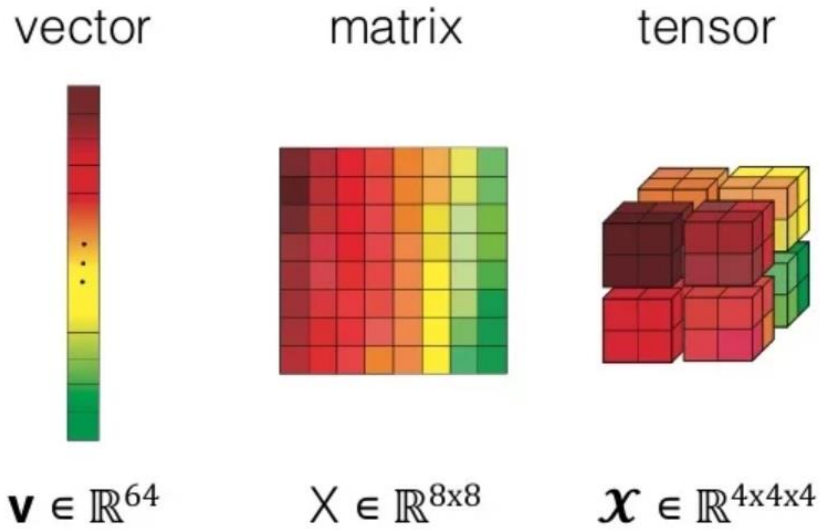


Итоговая модель для NER



Tensorflow (1)

https://www.tensorflow.org/



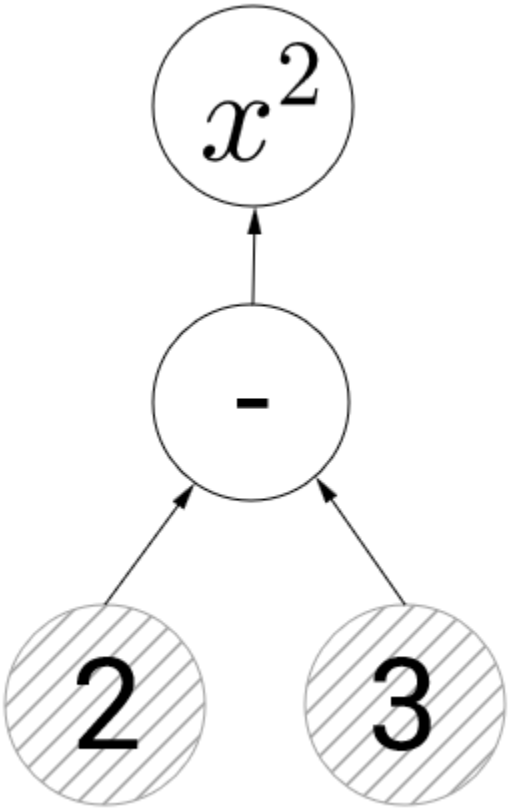
TensorFlow

- Библиотека для тензорных вычислений с открытым исходным кодом (Python, C++ и др.), разработанная Google
- Тензор = многомерный массив (скаляр (массив с одним элементом), вектор, матрица, массивы с размерностями > 2)
- Параллельная обработка на:
  - CPU, GPU
  - Нескольких CPU, GPU, серверах
- Библиотека для машинного обучения (мат. операции, алгоритмы оптимизации, мат. модели, нейросетевые архитектуры и др.)

Tensorflow (2)

- Концепция работы Tensorflow:
  - Любые данные в библиотеке – это тензоры (входные, выходные, промежуточные данные)
  - Вычисления проводятся внутри сессии
  - Перед началом работы сессии строится граф вычислений (граф математических операций)
  - При этом определяются точки ввода данных (placeholders)
  - Граф запускается на вычисление в рамках сессии, при этом всем placeholders передаются значения извне
  - Результат работы (тензор) можно извлечь из tensorflow и дальше работать с ним как с обычным многомерным массивом
  - Переменные внутри графа «живут» только пока «жива» сессия

Tensorflow (3)



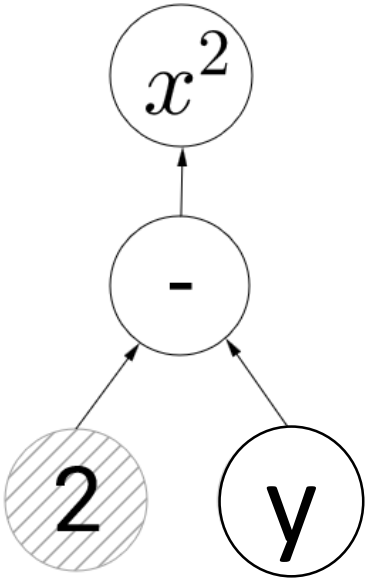

```
import tensorflow as tf

c1 = tf.constant(2.0)
c2 = tf.constant(3.0)
x = c1 - c2
x_sqrd = tf.square(x)

with tf.Session() as sess:
    print(sess.run(x_sqrd))
```

```
1.0
```

Tensorflow (4)



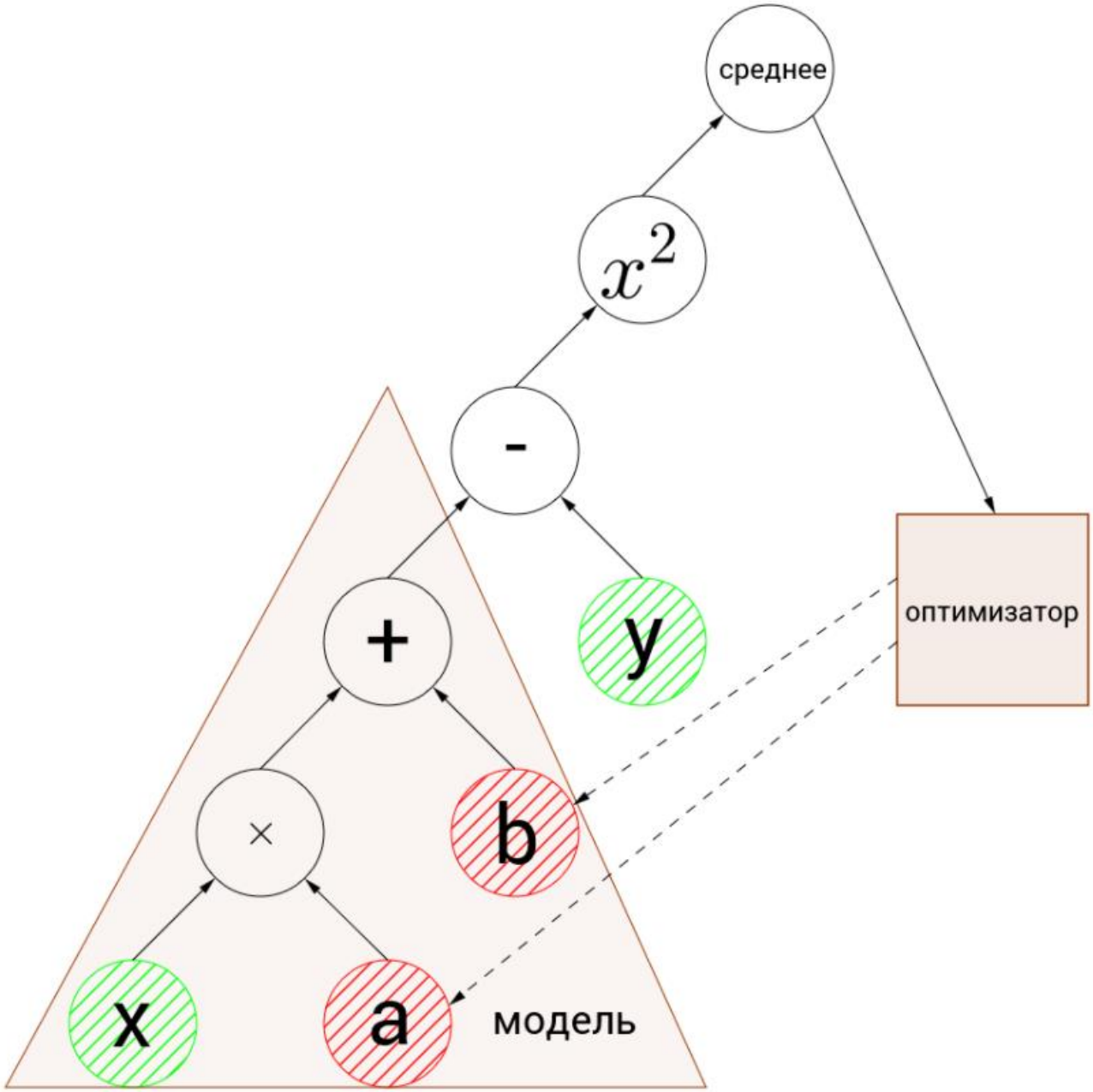

```
import tensorflow as tf

c1 = tf.constant(2.0)
y = tf.placeholder(tf.float32,
                   shape = [1],
                   name = 'y')
x = c1 - y
x_sqrd = tf.square(x)

with tf.Session() as sess:
    print(sess.run(x_sqrd,
                   feed_dict = {y : [10.]}))
```

```
[ 64.]
```

Машинное обучение в Tensorflow



Live demo

- Для решения задачи потребуются
  - Python 3.6
  - Пакеты: numpy, scipy, sklearn, tensorflow, genism и др.
  - Jupyter IDE – web-среда разработки для задач анализа данных
- Все это можно скачать и установить в 2 шага:
  - Установить Docker (среда управления виртуальными и псевдовиртуальными машинами)
    - Windows: https://www.docker.com/docker-windows
    - Mac: https://www.docker.com/docker-mac
    - Debian Linux: https://www.docker.com/docker-debian
  - Запустить команду:
    - docker run -ti --rm -v `pwd`:/notebook -p 8888:8888 windj007/jupyter-keras-tools
    - Jupyter поднимется по адресу http://localhost:8888
- Подробности: https://hub.docker.com/r/windj007/jupyter-keras-tools/
- Jupyter notebook с примером: http://nlp.isa.ru/hse/crf-lstm/

Заключение

- На сегодняшний день NLP и машинное обучение тесно связаны
- Быстро развиваются направления:
  - Deep learning (DL) (глубокие нейронные сети)
  - Embedding (векторные представления конструкций текста)
- Важным направлением в NLP является sequence modeling
  - С применением условно случайных полей
  - Рекуррентных нейронных сетей (LSTM)
- Хорошие инструменты, с которых можно начать применять глубокое обучение:
  - Tensorflow
  - Keras
- Чтобы быстро и просто начать использовать ML и DL (в том числе и на GPU) можно воспользоваться контейнером
  - Docker + https://hub.docker.com/r/windj007/jupyter-keras-tools/

Полезные материалы

- Машинное обучение:
  - http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf
- Глубокое обучение:
  - http://www.machinelearning.ru/wiki/images/7/71/Voron-ML-DeepLearning-slides.pdf
- Условно-случайные поля (CRF):
  - http://homepages.inf.ed.ac.uk/csutton/publications/crftut-fnt.pdf
- GitHub LSTM+CRF:
  - https://github.com/guillaumegenthial/sequence_tagging
- Статьи про модель типа LSTM+CRF:
  - https://arxiv.org/pdf/1603.01360.pdf
  - https://arxiv.org/pdf/1508.01991
  - https://arxiv.org/pdf/1709.09686 (Russian NER)
- Jupyter notebook с разобранным примером:
  - http://nlp.isa.ru/hse/crf-lstm/

Литература

- Mikolov T. et al. Distributed representations of words and phrases and their compositionality //Advances in neural information processing systems. – 2013. – С. 3111-3119
- Pennington J., Socher R., Manning C. Glove: Global vectors for word representation //Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). – 2014. – С. 1532-1543
- Hochreiter S., Schmidhuber J. Long short-term memory //Neural computation. – 1997. – Т. 9. – №. 8. – С. 1735-1780
- Lafferty J., McCallum A., Pereira F. C. N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. – 2001