Рекуррентные нейронные сети

Семинар 19

Анализ текста. Использование

- Классификация текста
- Определение эмоциональной окраски (тональность) текста
- Извлечение сущностей из текста
- Реферирование/аннотирование
- Генерация текста
- Автоматический перевод
- Чат-боты

Анализ текста как последовательности

Анализ текста полносвязной сетью

- Все токены текста поступают на вход каждому нейрону
- Токены анализируются изолированно друг от друга

Проблемные тексты для полносвязной сети:

- Overall, the movie is not bad and has entertainment value.
- Unfortunately, the movie is not so good.
- Ice cream (мороженое, ice – лед, cream – крем, сливки)

Необходимо анализировать текст как последовательность токенов

- Порядок слов/символов/предложений в тексте имеет большой смысл
- Нужны специальные архитектуры нейронных сетей для анализа последовательностей

Операция с текстом. Токенизация

- Символы

Буквы, цифры, знаки препинания и т.п.

- Слова
- Предложения
- Методы извлечения признаков из текста

N-граммы (последовательности слов длиной от 1 до N)

Мешок слов (множество всех слов)

N-граммы

Биграммы – это последовательности из двух слов (I want, want to, to, go, go to, to the…), триграммы – последовательности из трех слов (I want to, want to go, to go to…).
Подсчитываем вероятность следующего слова или последовательности слов. Такие подсчеты называются языковыми моделями.
Как же рассчитать P(w)? Например, вероятность предложения P(I, found, two, pounds, in, the, library). Для этого нам понадобится цепное правило, которое определяется так:

$$\mathrm{P}(A, B) = \mathrm{P}(B \mid A)\mathrm{P}(A)$$

$$\mathrm{P}(A_1, A_2, A_3, A_4) = \mathrm{P}(A_4 \mid A_3, A_2, A_1) \cdot \mathrm{P}(A_3 \mid A_2, A_1) \cdot \mathrm{P}(A_2 \mid A_1) \cdot \mathrm{P}(A_1)$$

N-граммы

P(I, found, two, pounds, in, the, library) =

P(I) *
P(found | I) *
P(two | I found) *
P(pounds | I found two) *
P(in | I found two pounds) *
P(the | I found two pounds in) *
P(library | I found two pounds in the)

Таким образом, мы можем оценить совместную вероятность всей цепочки, перемножив условные вероятности. Однако мы не можем рассчитать точную вероятность слова при условии длинной последовательности предшествующих слов. Это потому, что возможных последовательностей очень много, а в наших данных просто может не оказаться этих выражений. Поэтому, вместо того, чтобы рассчитывать вероятность слова с учетом всех предшествующих слов, мы можем аппроксимировать вероятность, упростив ее. В этом заключается смысл цепей Маркова, с помощью которых мы можем предсказать вероятность элемента последовательности, не учитывая слишком широкий контекст.
Например, марковская модель первого и второго порядка:

$$P(\text{library} \mid \text{I found two pounds in the}) \approx P(\text{library} \mid \text{the})$$

$$P(\text{library} \mid \text{I found two pounds in the}) \approx P(\text{library} \mid \text{in the})$$

Операция с текстом. Векторизация

- Представление текста в виде чисел (Word2Vec, GloVe, FastText, RusVectores, RUSSE)
- Числовое кодирование

Для каждого токена свой код (ASCII, UTF-8 и т.п.)

- One hot encoding

Вектор по одному символу на каждый возможный токен

Все элементы вектора 0, кроме того, который соответствует токену

- Плотные векторные представления (embedding)

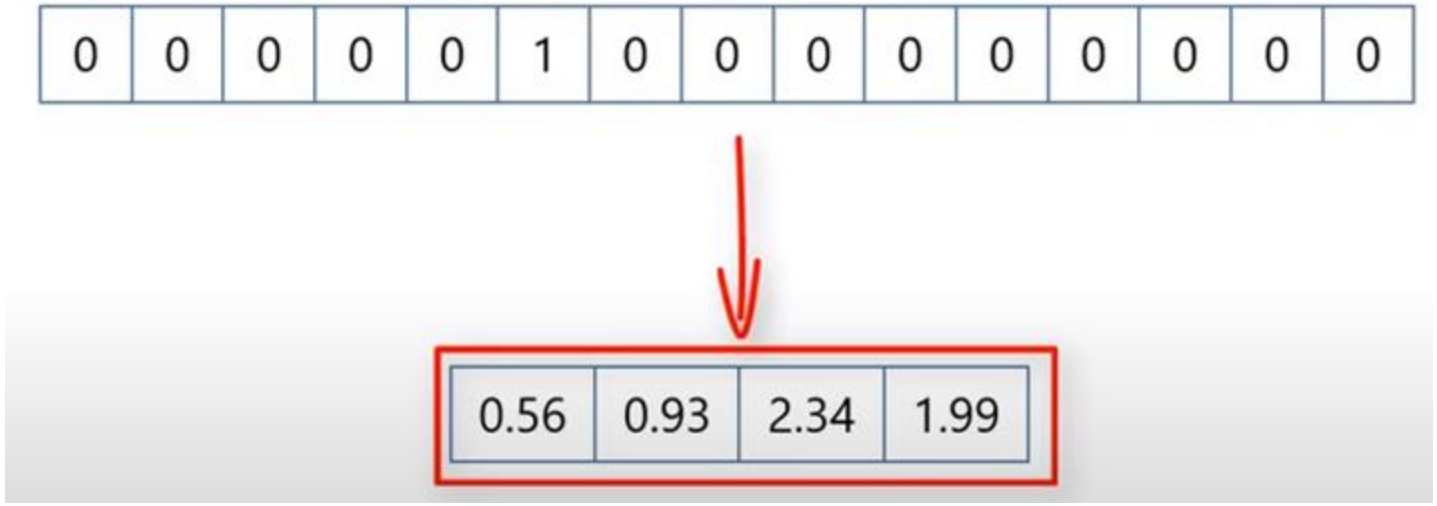
Каждому токену сопоставляется вектор

Размерность вектора ниже, чем у one hot encoding

One hot encoding

- Ограничиваем максимальное количество используемых слов
- Длина вектора для каждого слова равна максимальному количеству слов
- Значения элементов вектора
- 0 – если слово не встречается в тексте
- 1 – если встречается

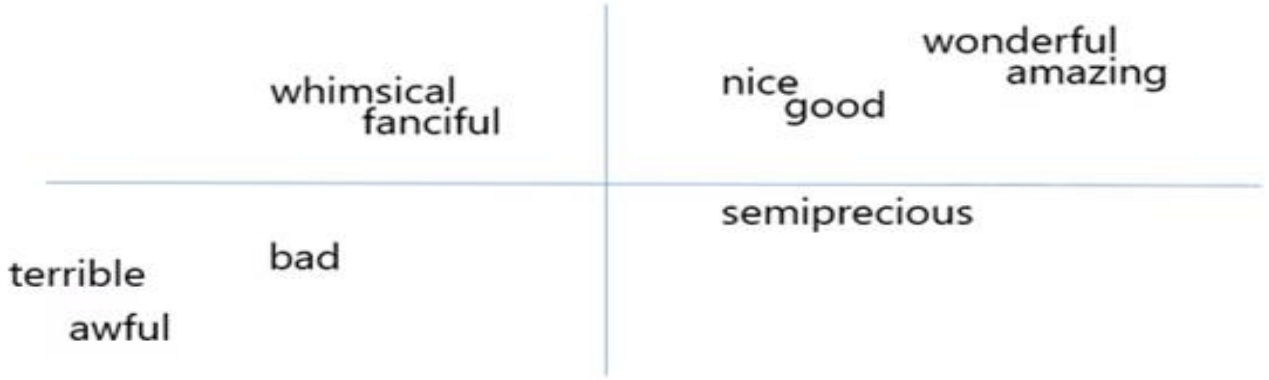
Разреженные и плотные векторы



Обучение плотных векторных представлений

В нейронных сетях плотное векторное представление слов определяется в процессе обучения

- На первом этапе элементы векторов инициализируются случайными числами
- Изменение значений векторов с помощью метода обратного распространения ошибки до того как векторное представление слов подходит для решения задачи



Итоги

Формат входных данных в нейронную сеть

- Числа

Токенизация текста

- Символы, слова, предложения

Векторизация текста

- Коды токенов, one hot encoding, плотные векторные представления

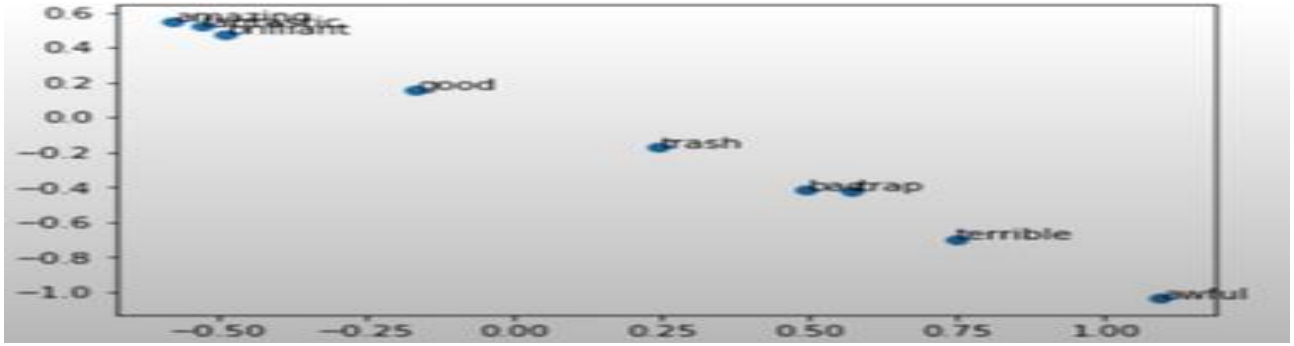
Преобразование текста в набор чисел

- Комбинация методов токенизации и векторизации
- Разные методы подходят для разных типов задач

Проблема в работе полносвязной нейронной сети

Анализ текста полносвязной сетью

- Все токены текста поступают на вход каждому нейрону
- Токены анализируются изолированно друг от друга
- У слова есть вес. Чем больше слов с позитивной окраской, тем позитивней текст



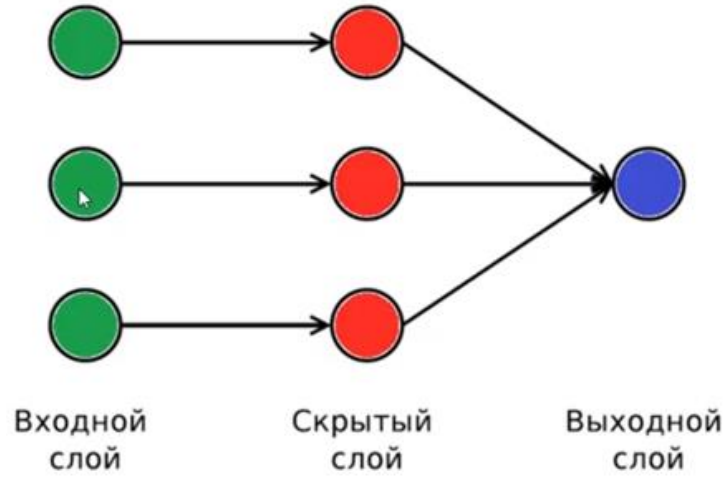
Проблемные тексты для полносвязной сети

- The movie is not bad…
- The movie is not so good

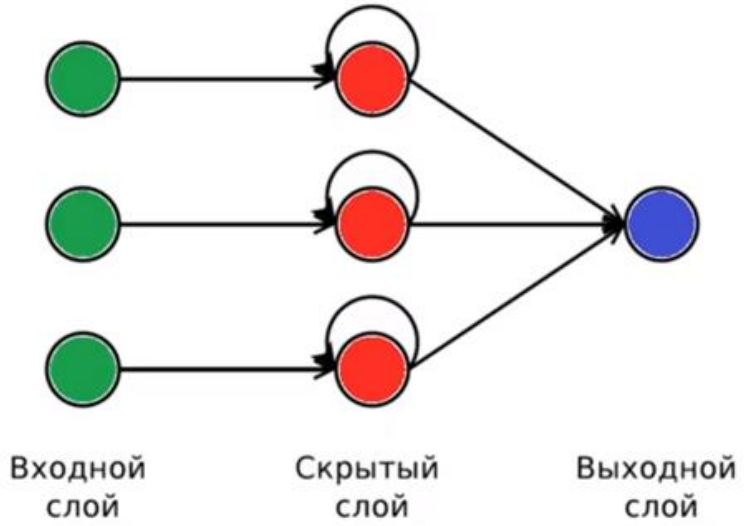
Необходимо анализировать текст не как набор токенов, а как последовательность токенов

- Порядок слов/предложений в тексте имеет большой смысл
- Нужны специальные архитектуры нейронных сетей для анализа последовательностей

Типы нейронных сетей

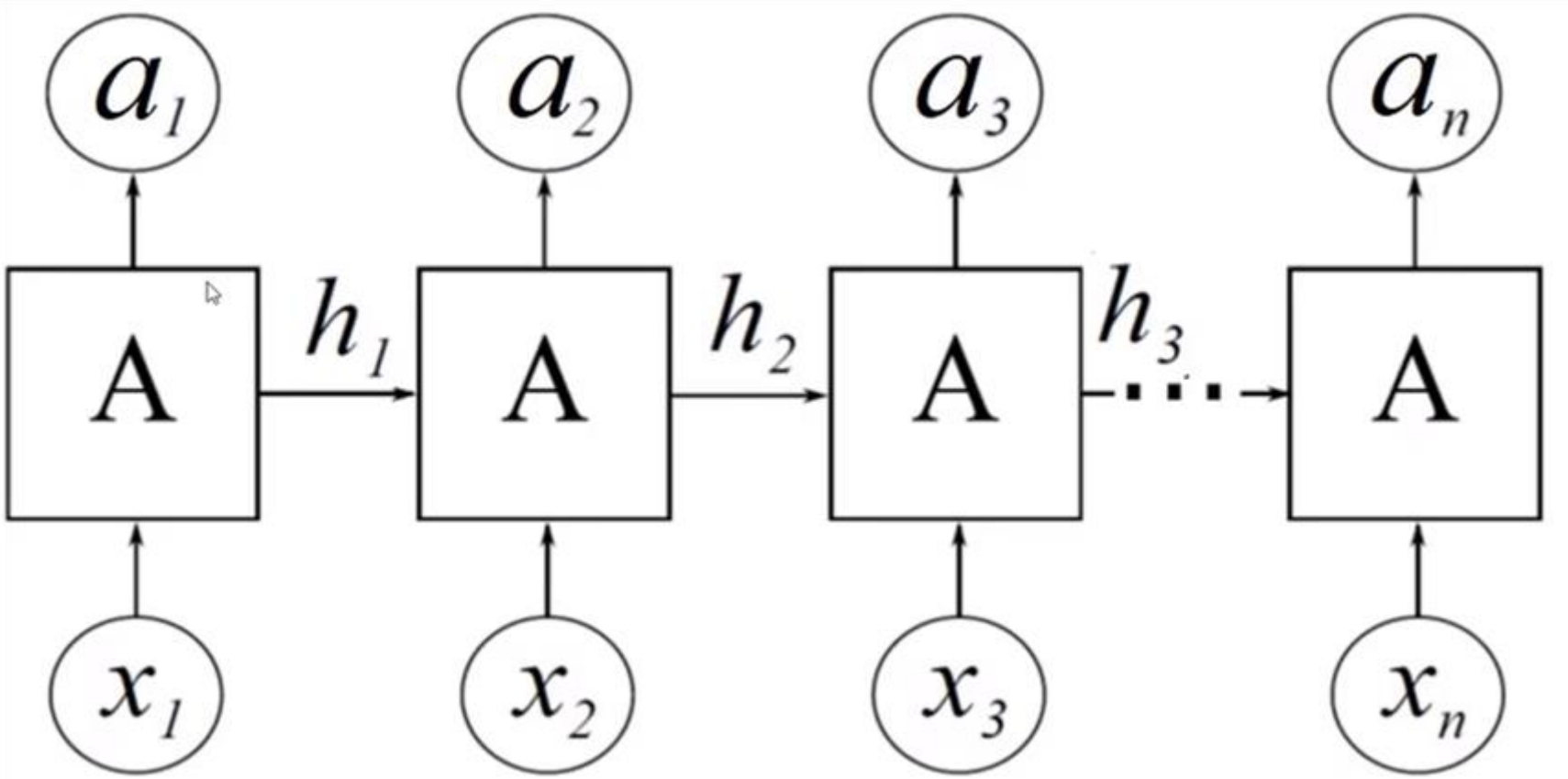


Сети с прямым распространением сигнала
(feedforward networks)



Рекуррентные сети
(recurrent networks)

Разворачивание рекуррентной нейронной сети во времени
1 слой = 1 слово

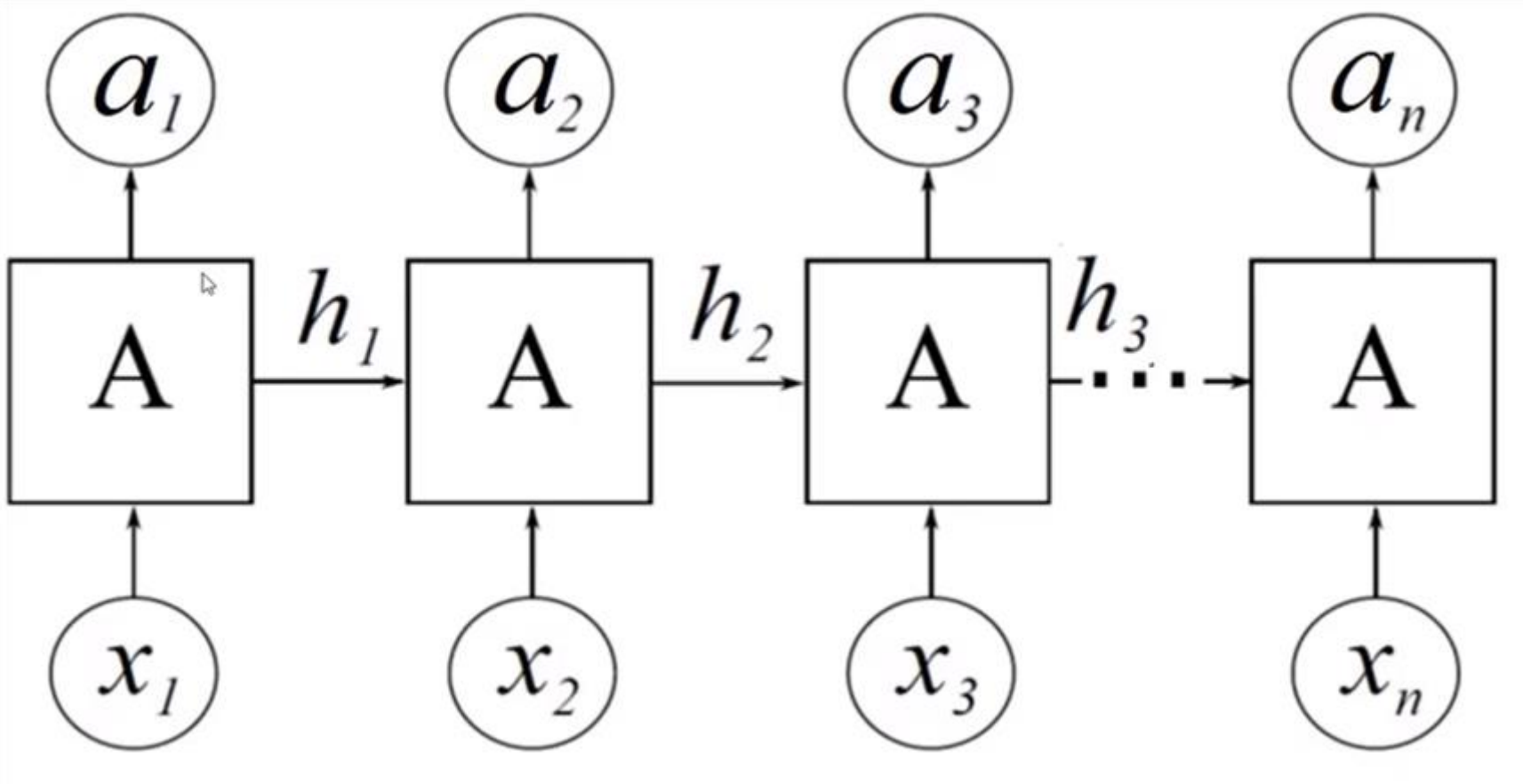


A – RNN
X – входной элемент последовательности
a – выходной элемент последовательности
h – значение со скрытым состоянием с предыдущего этапа

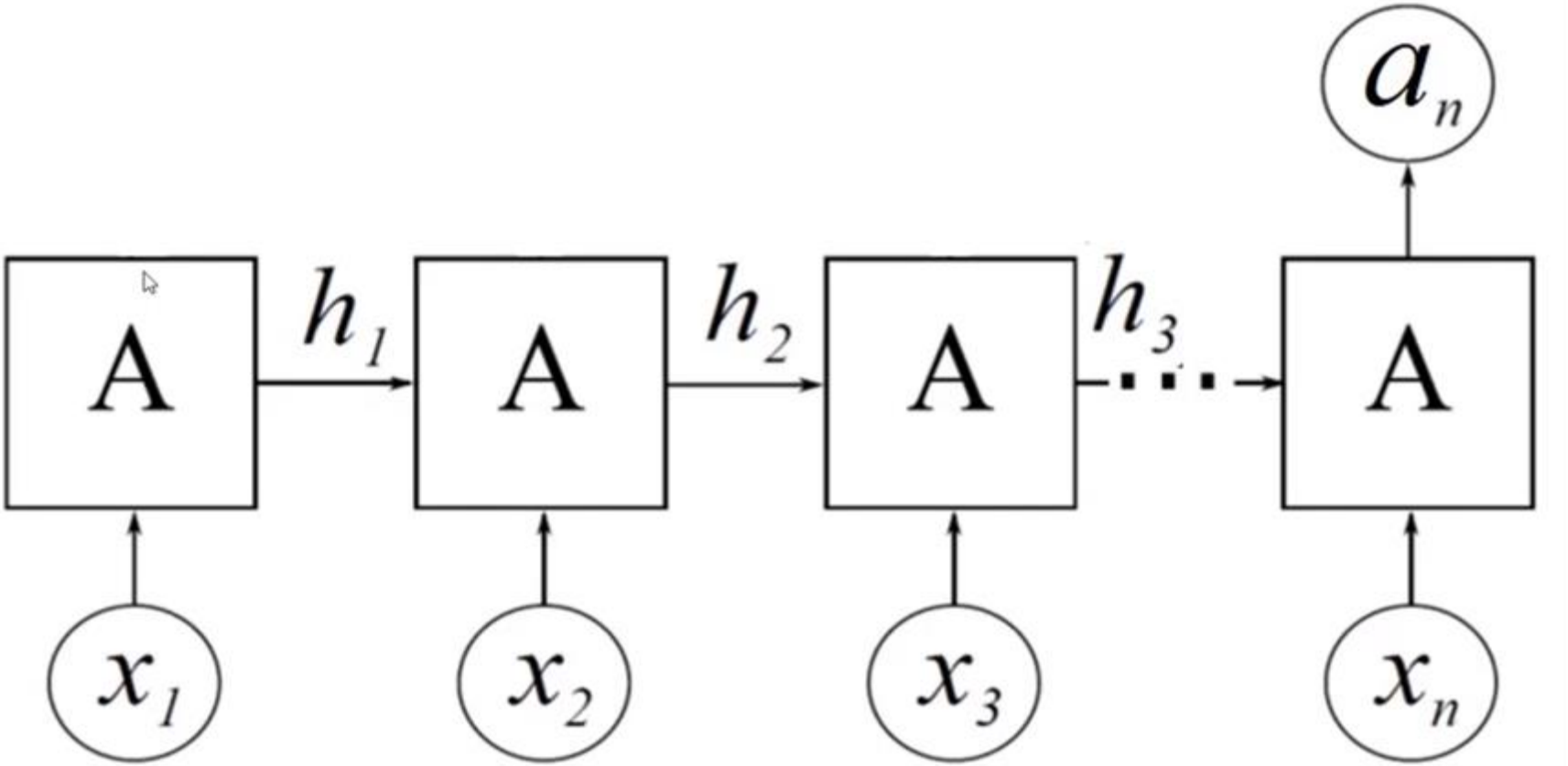
Рекуррентные нейросети в Keras

```
model = Sequential()
model.add(Embedding(input_dim=max_words,
                    output_dim=50,
                    input_length=maxlen))
model.add(SimpleRNN(8))
model.add(Dense(1, activation='sigmoid'))
```

Режим sequence to sequence (автоперевод, генерация текста)



Режим sequence to vector (классификация)



Режимы работы рекуррентных сетей в Keras

```
model = Sequential()
model.add(Embedding(input_dim=max_words,
                    output_dim=50,
                    input_length=maxlen))
model.add(SimpleRNN(16,
                    return_sequences=True))
model.add(SimpleRNN(8))
model.add(Dense(1, activation='sigmoid'))
```

Обучение рекуррентных нейронных сетей

Для обучения рекуррентных нейронных сетей используется разворачивание во времени и обратное распространение ошибки

- Количество слоев (100 слов = 100 слоев) в развернутой сети зависит от длины последовательности входных данных

Проблема исчезающего градиента

- При передаче от слоя к слою сигнал об изменении весов уменьшается
- Сеть с большим количеством слоев (100 и более) сложно обучить

Сложные архитектуры нейронных сетей

- LTSM
- GRU

Итоги

Рекуррентные нейронные сети

- Сети с циклами

Анализ последовательностей

- Тексты, речь, временные ряды

Проблемы рекуррентных нейронных сетей

- Обучение требует длительного времени
- Проблема исчезающего градиента
- Ограниченная «длительность» запоминания предыдущей информации (до 10 шагов)

Пути решения проблем

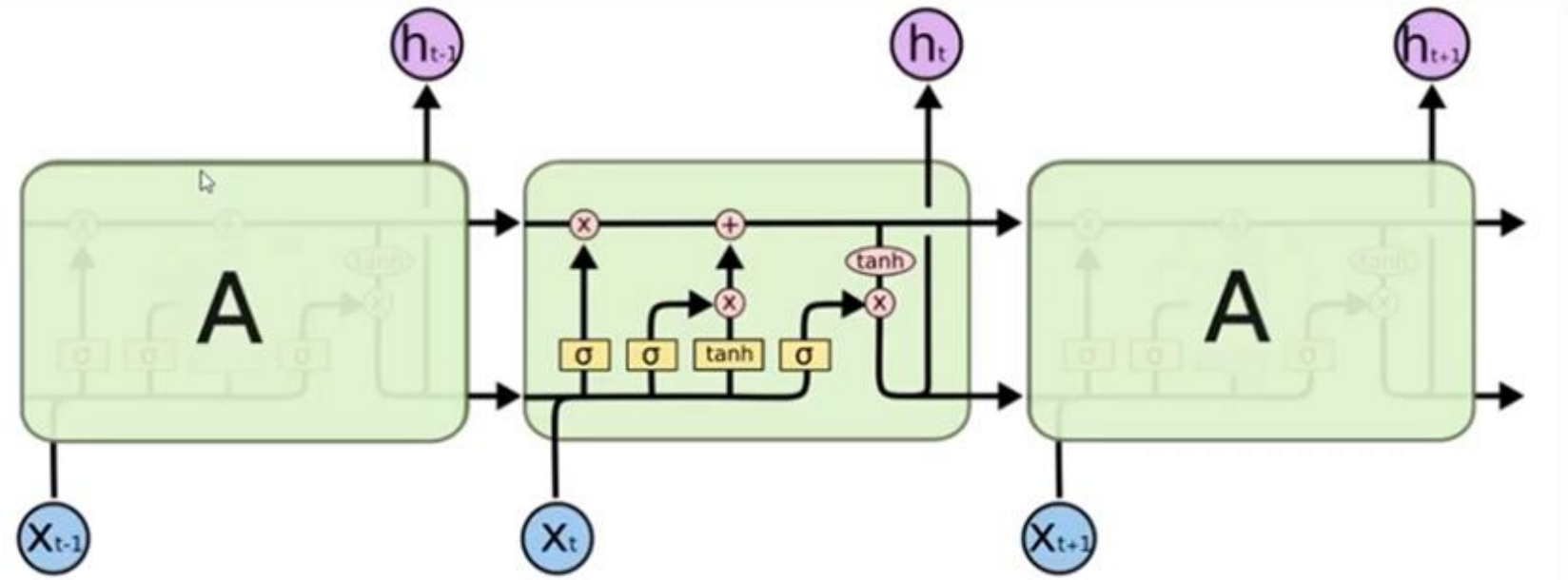
- Более совершенные архитектуры рекуррентных сетей LSTM и GRU
- Одномерные сверточные нейронные сети

Задание

- Практика с рекуррентными нейронными сетями. Анализ тональности текста
- https://colab.research.google.com/drive/1ThXlQsO3KycFQd408k12qv65NY4FLat_#scrollTo=xGDJkdINNWVh

Сети LSTM и GRU

Сложные нейросети. Элементом сети является не один нейрон или слой из нейронов,
а целый набор слоев, которые взаимодействуют друг с другом по определенным правилам.



Understanding LSTM Networks

- https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Как понять LSTM сети

- https://alexsosn.github.io/ml/2015/11/17/LSTM.html

Сети LSTM в Keras

```
model = Sequential()
model.add(Embedding(input_dim=max_words,
                    output_dim=50,
                    input_length=maxlen))
model.add(LSTM(8))
model.add(Dense(1, activation='sigmoid'))
```

Сети GRU

```
model = Sequential()
model.add(Embedding(max_words, 8, input_length=maxlen))
model.add(GRU(32))
model.add(Dense(1, activation='sigmoid'))
```

Задание

- Практика с рекуррентными нейронными сетями GRU, LSTM. Анализ тональности текста
- https://colab.research.google.com/drive/17G_mv0KhuerZT-ri0tJky53dEz5rfyJJ#scrollTo=a19UCeAMDhQK