Применение модели глубокого обучения sequence-to-sequence в задаче формирования аннотаций

TensorFlow/textsum

Ларюшина Юлия
Шашкин Павел
15МАГПМИ

План работы

- Изучение системы машинного обучения TensorFlow
- Установка и настройка библиотеки
- Сбор данных для обучения
- Решение распространенных проблем
- Создание туториала для разработчиков
- Обучение модели textsum
- Создание интерфейса для взаимодействия с моделью

TensorFlow / общая информация

- Граф потока данных, представляющий вычисления
- Вершины - операции (operation)
- Ребра – тензоры (tensor)
- Вычисления реализуются в рамках сессии (session)
- Вычисления выполняются на устройствах (device) (CPU или GPU)

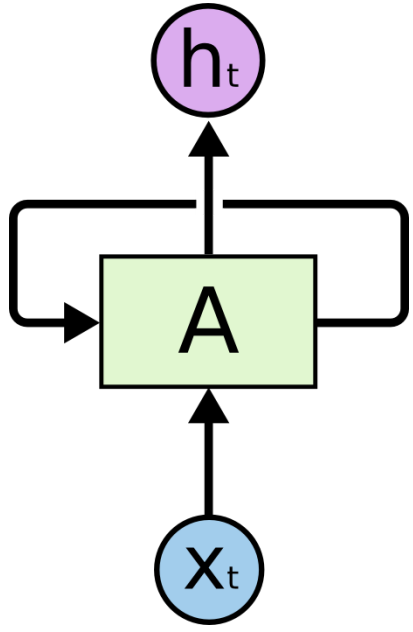
TensorFlow / общая информация

- Имеет api для Python, для R – в разработке
- TensorFlow выполняет вычисления с помощью высоко оптимизированного C++, а также поддерживает нативный API для C и C++

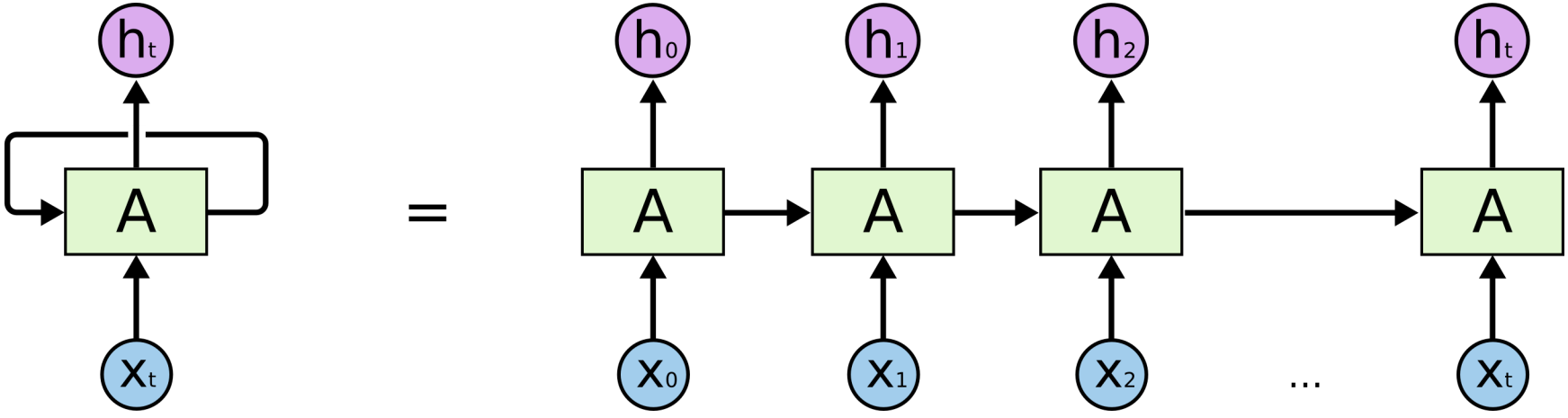
TensorFlow / простой пример

```
import tensorflow as tf
import numpy as np
matrix1 = 10 * np.random.random_sample((3, 4))
matrix2 = 10 * np.random.random_sample((4, 6))
tf_matrix1 = tf.constant(matrix1)
tf_matrix2 = tf.constant(matrix2)
tf_product = tf.matmul(tf_matrix1, tf_matrix2)
sess = tf.Session()
result = sess.run(tf_product)
sess.close()
```

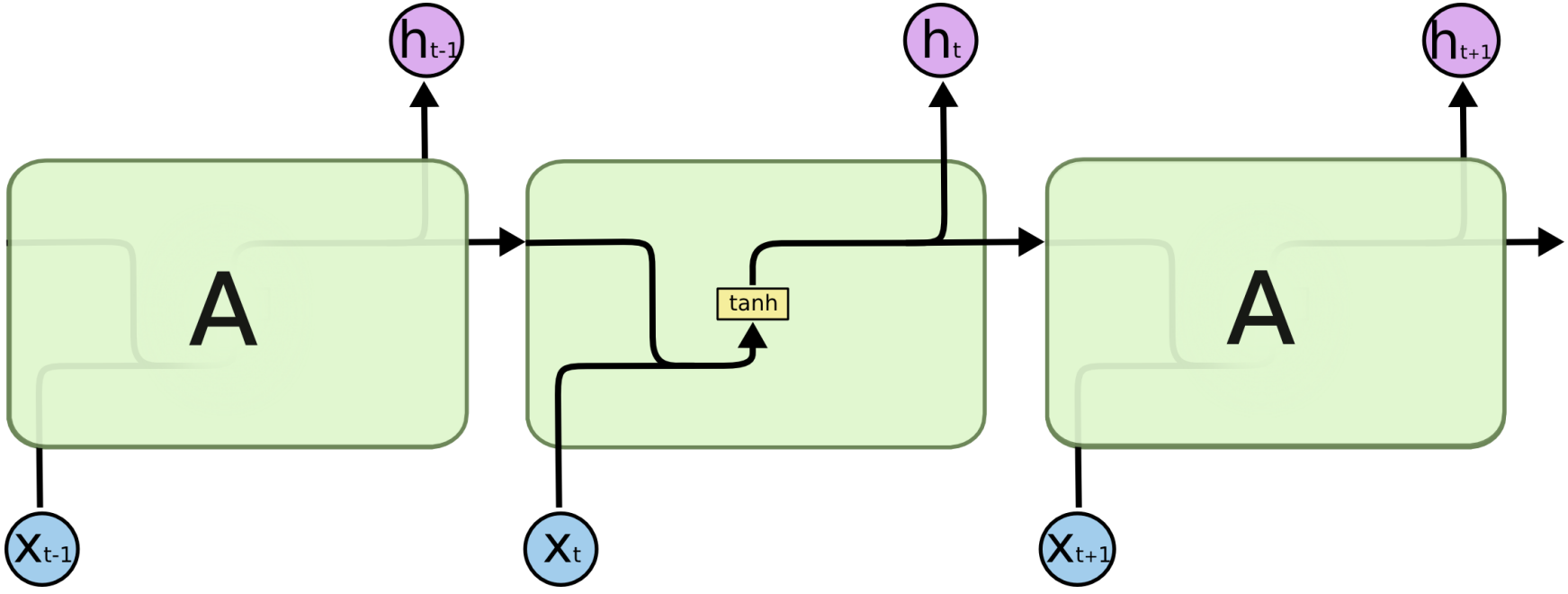
RNN



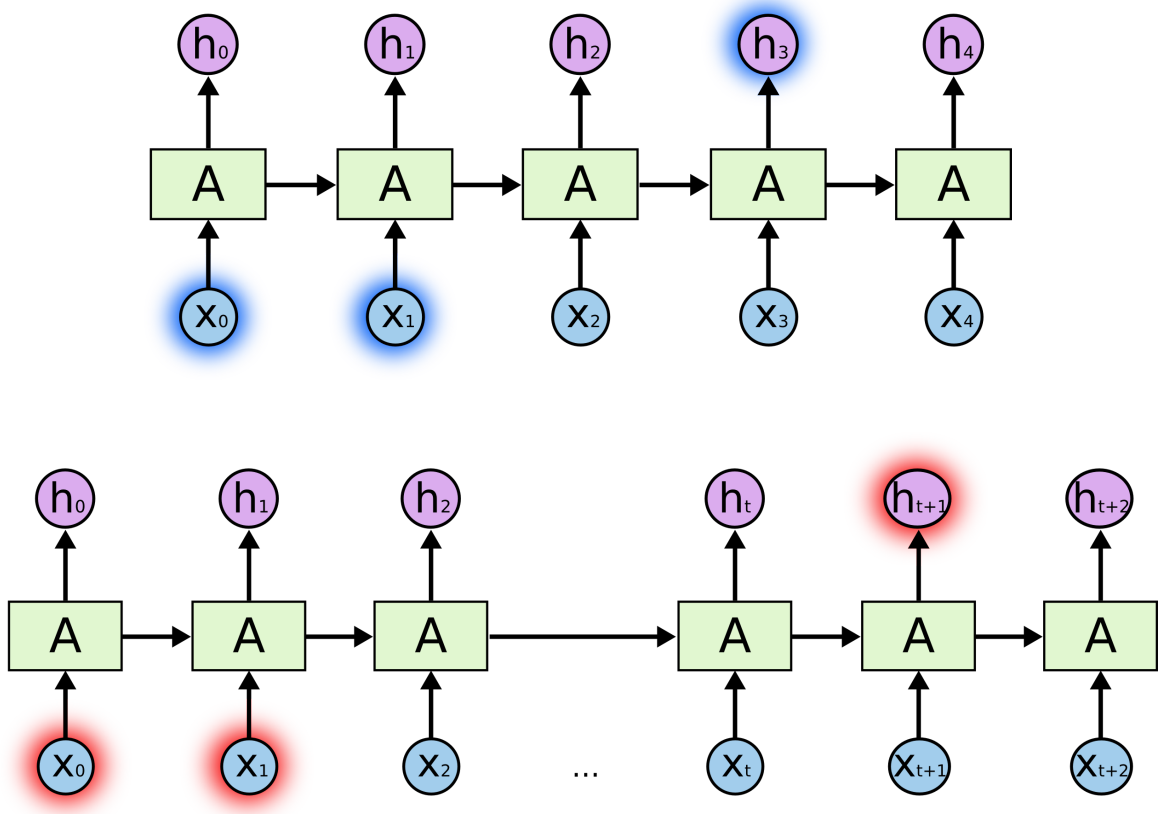
RNN



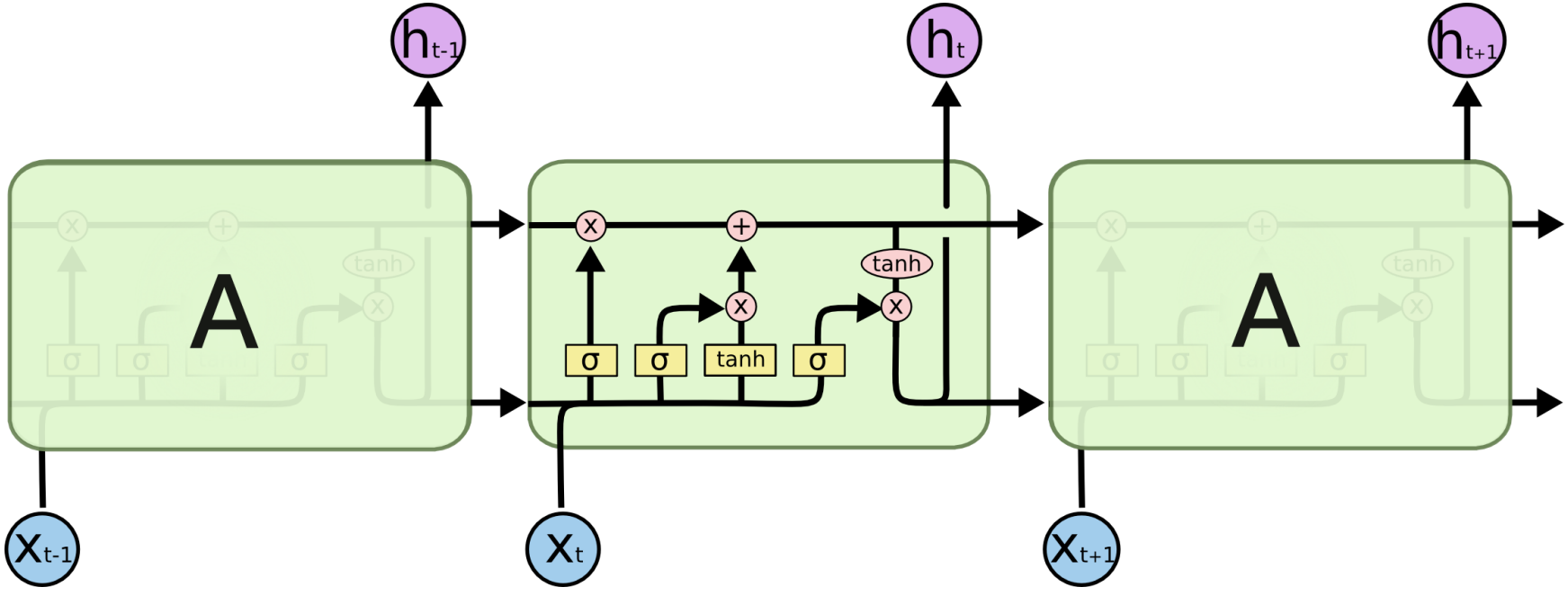
RNN традиционная



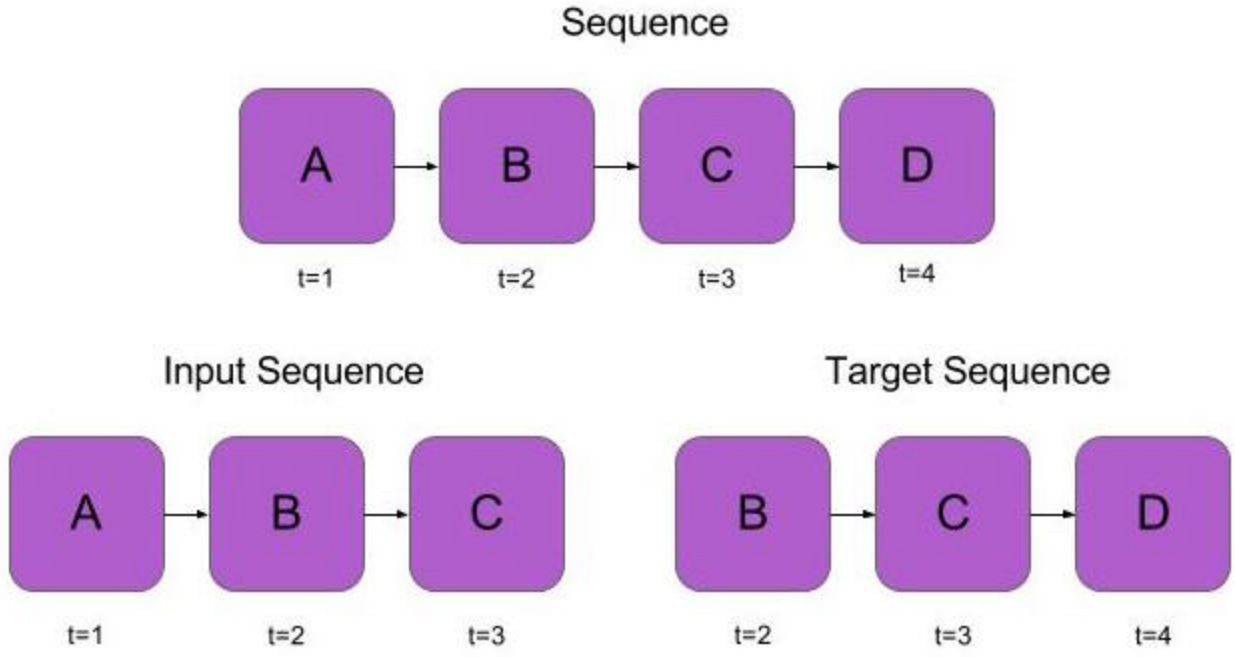
RNN проблема зависимостей



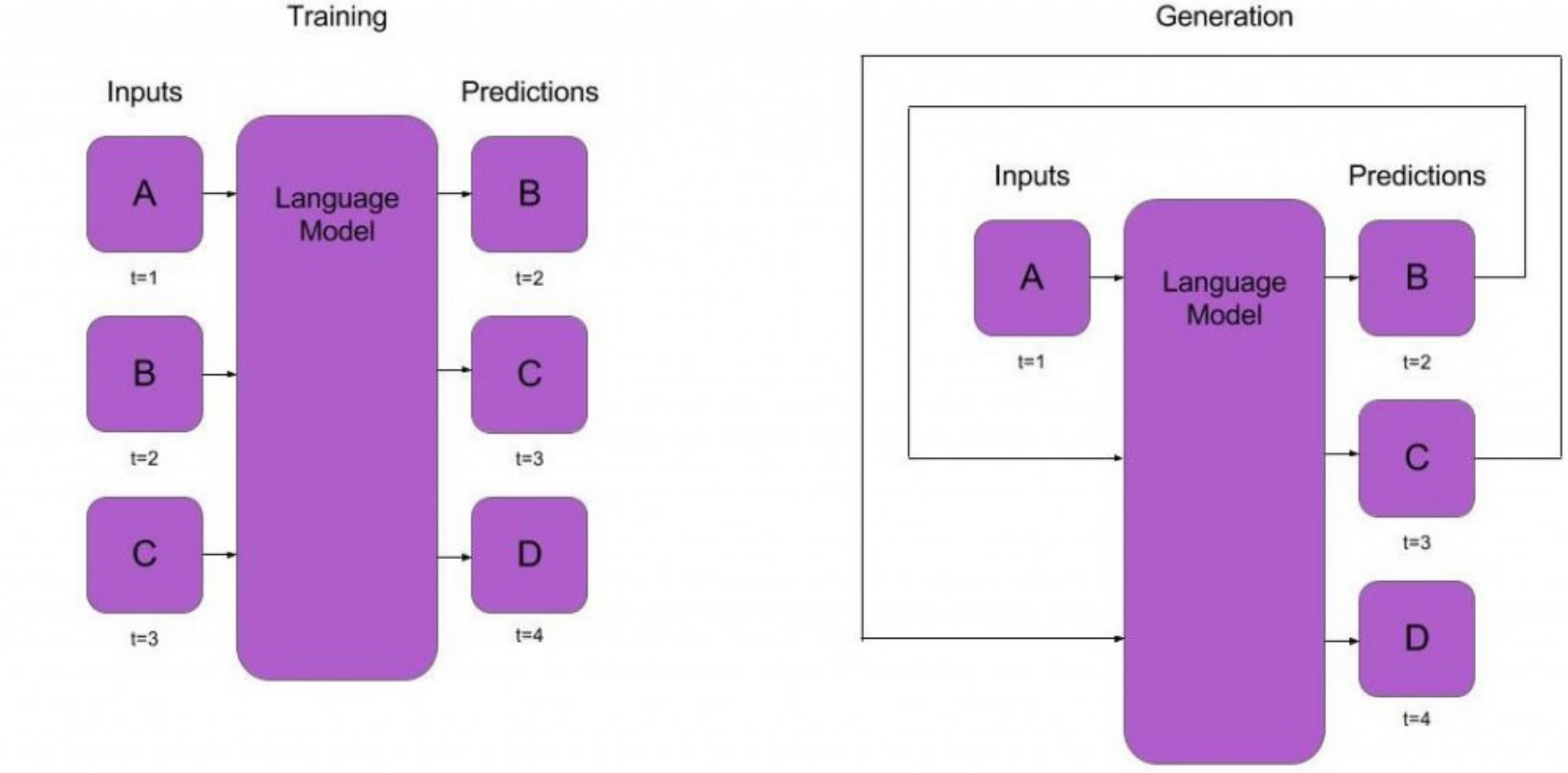
LSTM



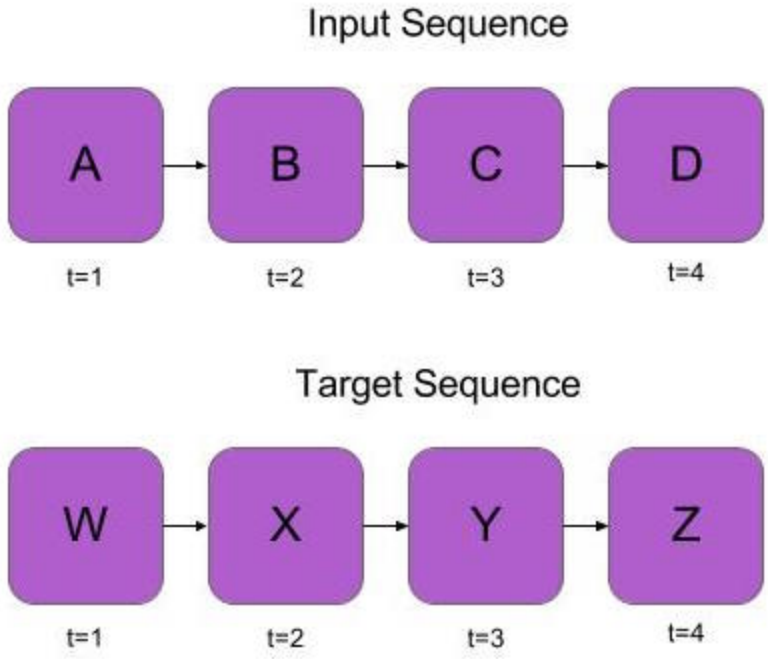
Базовая задача



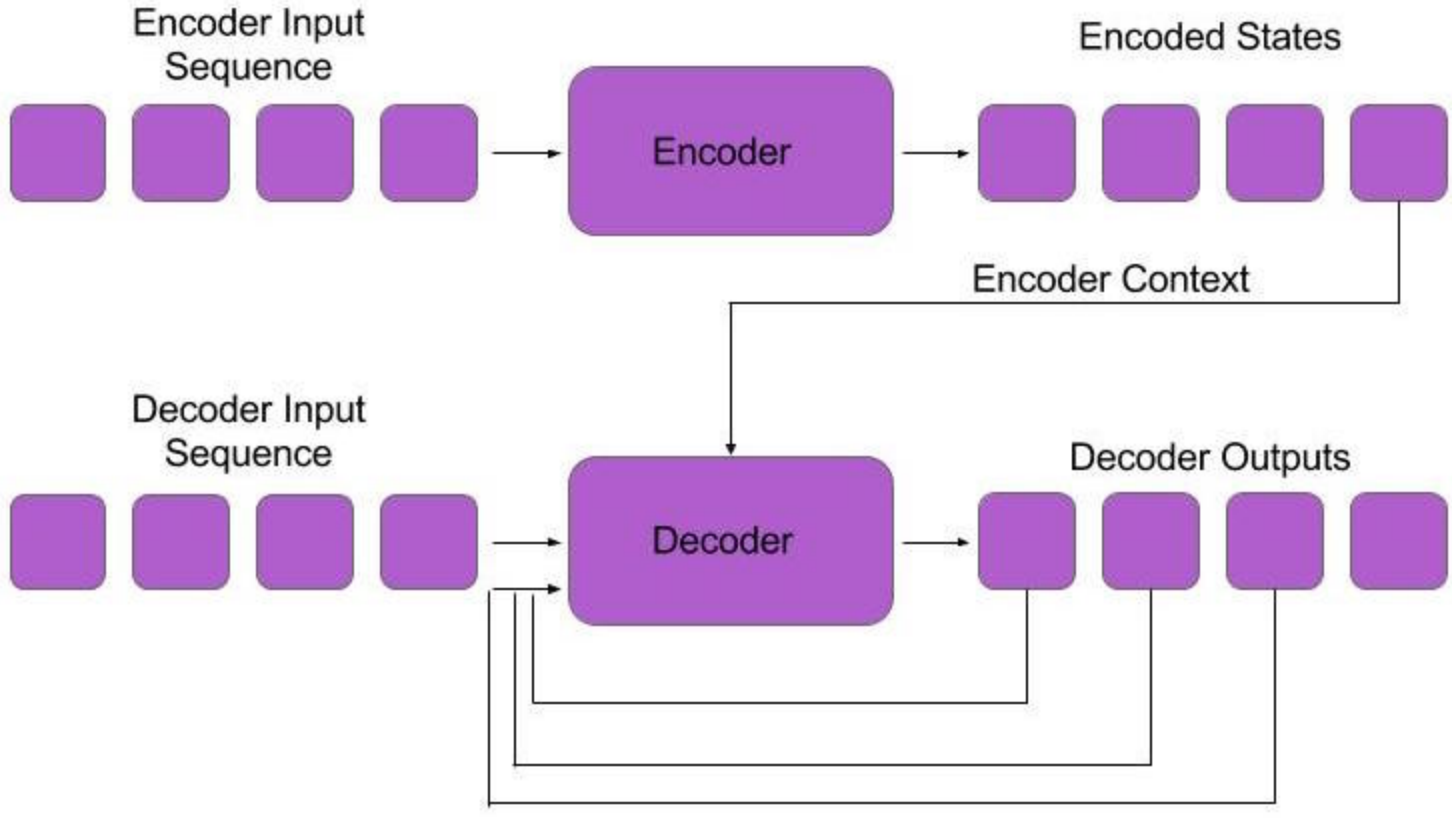
Модель для решения задачи



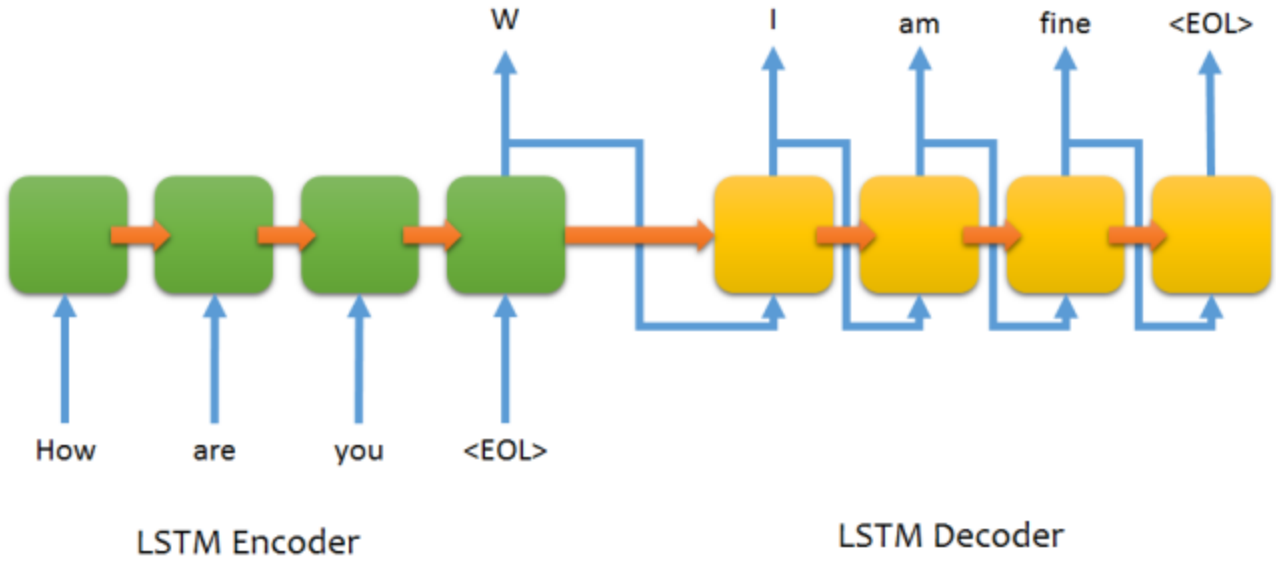
Задача для sequence-to-sequence



Sequence-to-sequence модель

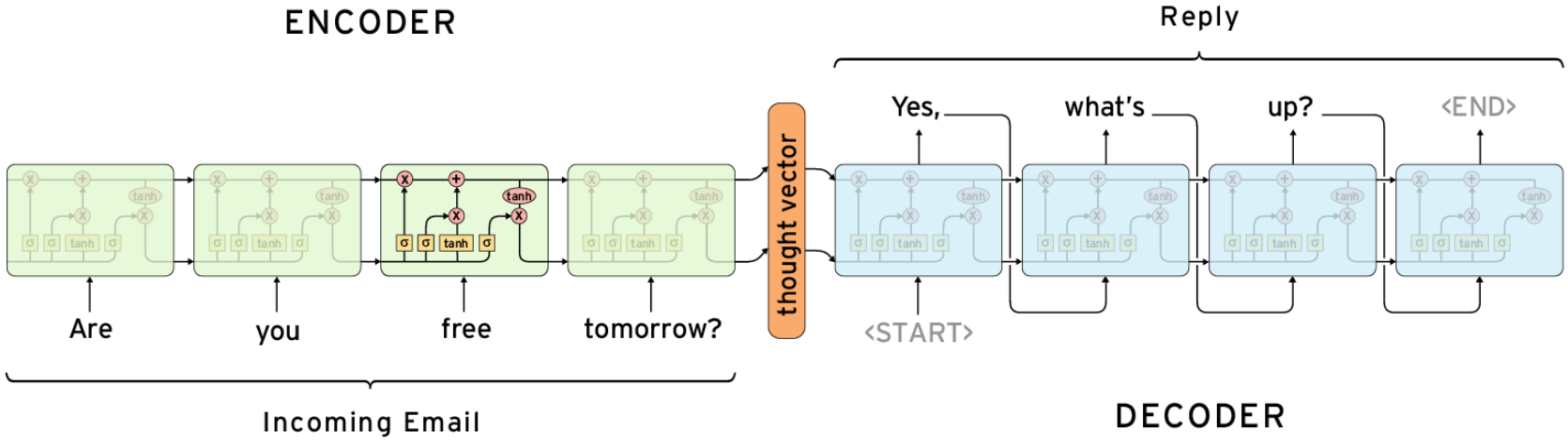


Sequence-to-sequence модель

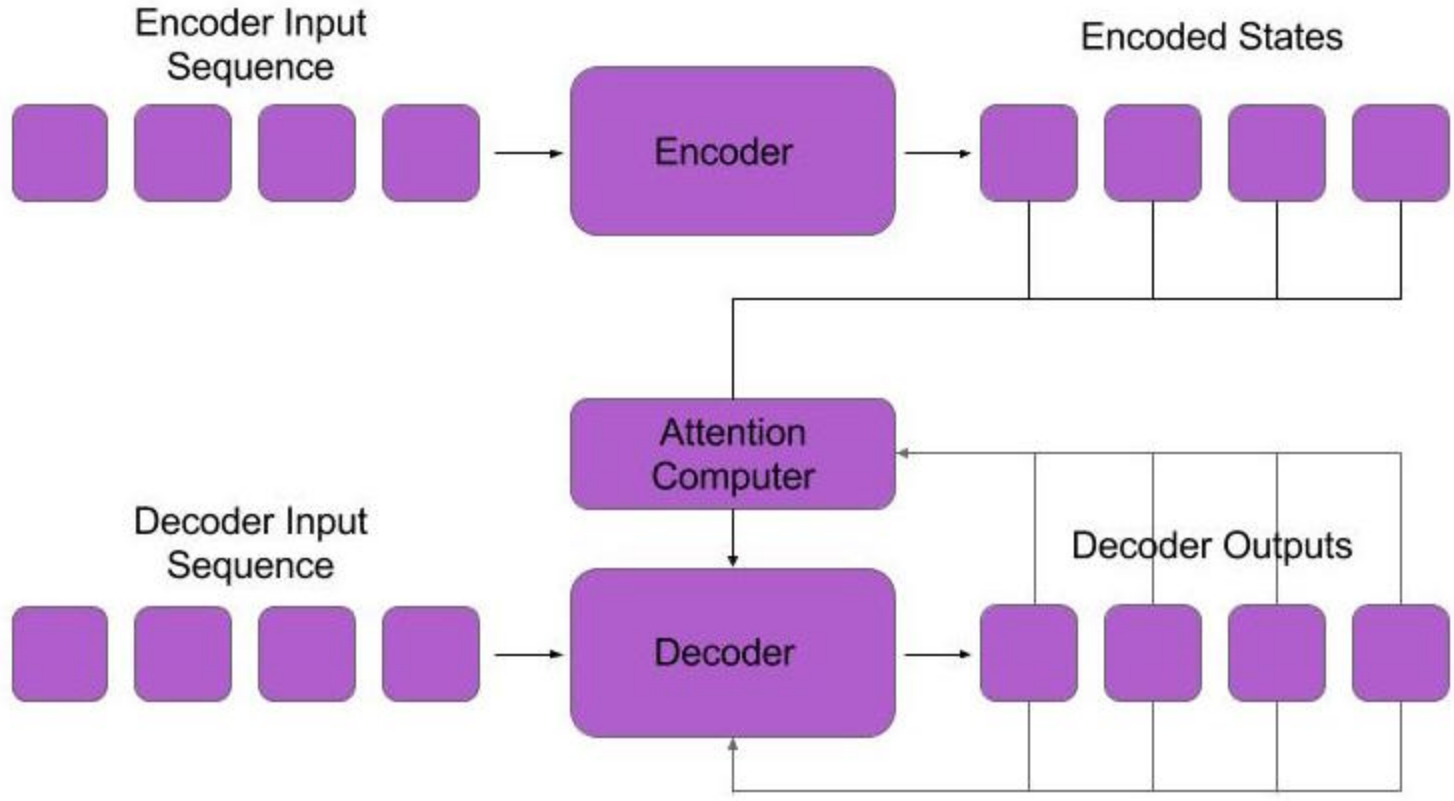


- Каждый прямоугольник - ячейка RNN (GRU или LSTM)
- Encoder и decoder используют различный набор параметров

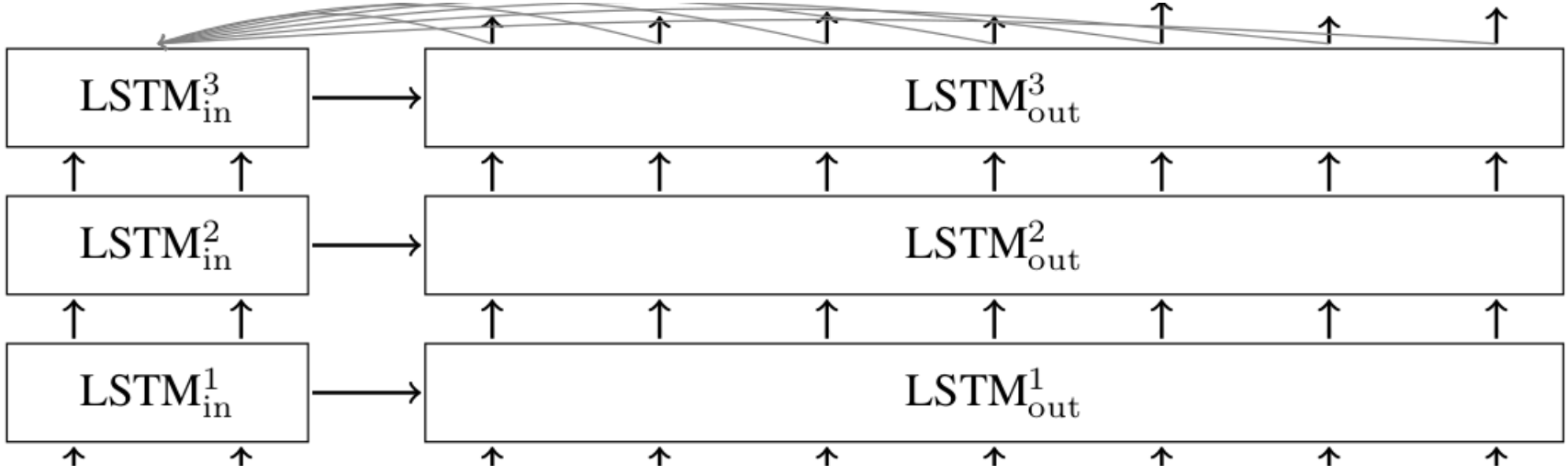
Sequence-to-sequence модель



Sequence-to-sequence with attention модель



Sequence-to-sequence модель / attention mechanism



Sequence-to-sequence / преобразования данных

- Padding
- Bucketing
- Word Embedding
- Reversing

Textsum / необходимые компоненты

Подробный туториал здесь

- TensorFlow 😄
- Bazel
- Python

TensorFlow / установка

- CPU only:
  - Python 2.7: export TF_BINARY_URL=https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-0.11.0rc0-cp27-none-linux_x86_64.whl
  - Python 2: pip install --ignore-installed --upgrade $TF_BINARY_URL

TensorFlow / установка

- Тестирование установки:

```
python
...
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
>>> print(sess.run(hello))
Hello, TensorFlow!
>>> a = tf.constant(10)
>>> b = tf.constant(32)
>>> print(sess.run(a + b))
42
>>>
```

Bazel / установка

- Bazel - открытая система для создания и тестирования приложений на разных платформах
- Bazel работает на Linux и OS X, его можно использовать для сборки и тестирования проектов на C++, Java, Python, а также он поддерживает Android и iOS приложения

После скачивания Bazel installer:

```
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java8-installer
$ sudo apt-get install pkg-config zip g++ zlib1g-dev unzip
$ chmod +x bazel-version-installer-os.sh
$ ./bazel-version-installer-os.sh --user
$ export PATH="$PATH:$HOME/bin"
```

Подготовка окружения

В директории workspace_sample содержится пример workspace для textsum.
Необходимо воссоздать его структуру и добавить пустой WORKSPACE файл.
Имеем:

- оригинальный textsum отсюда
- папку с "игрушечными" данными (training/data, testing/data, validation/data), пример реальных входных данных (text_data) и словарь (vocab)
- WORKSPACE файл (необходим для Bazel)

Подготовка окружения

Текстовый файл text_data содержит пример входных данных. Необходимо трансформировать данные в этот вид и в двоичный формат.

```
python data_convert_example.py --command text_to_binary --in_fi
```

Осталось создать приложение:

```
bazel build -c opt --config=cuda textsum/...
```

Формат данных

Для того, чтобы получить сколько-нибудь осмысленный результат, необходимо использовать формат Gigaword для тренировки.

Данные должны содержать следующее:

- Каждый элемент "article" начинается с предопределенного тэга (напр. `<article>`)
- Каждый элемент "abstract" начинается с предопределенного тэга (напр. `<abstract>`)
- Параграфы разделены тэгами `<p>` и `</p>`
- Предложения разделены тэгами `<s>` и `</s>`
- Абстракты и статьи разделены тэгами `<d>` и `</d>`
- Обучающие примеры разделены переносами строк или лежат в разных файлах

Формат данных

То есть сэмплы должны выглядеть как:

```
article=<d> <p> <s> here article1 sentence 1. </s> <s> here art
article=<d> <p> <s> here article2 sentence 1. </s> <s> here art
```

Формат данных

Также необходимо отметить, что имеет смысл создать свой собственный словарь, а не использовать полученный вместе с toy data.
Простейший скрипт для

- сбора данных,
- обработки данных
- формирования словаря.

"Игрушечный" пример

Для тренировки модели:

```
bazel-bin/textsum/seq2seq_attention \
  --mode=train \
  --article_key=article \
  --abstract_key=abstract \
  --data_path=data/training/data* \
  --vocab_path=data/vocab \
  --log_root=textsum/log_root \
  --train_dir=textsum/log_root/train
  --max_run_steps=N
```

N - максимальное количество этапов.

По умолчанию имеем 10000000, и результаты с CPU никогда таким образом не получим.

"Игрушечный" пример

Для валидации модели:

```
bazel-bin/textsum/seq2seq_attention \
  --mode=eval \
  --article_key=article \
  --abstract_key=abstract \
  --data_path=data/validation/data* \
  --vocab_path=data/vocab \
  --log_root=textsum/log_root \
  --eval_dir=textsum/log_root/eval
```

"Игрушечный" пример

Для тестирования модели:

```
bazel-bin/textsum/seq2seq_attention \
  --mode=decode \
  --article_key=article \
  --abstract_key=abstract \
  --data_path=data/test/data* \
  --vocab_path=data/vocab \
  --log_root=textsum/log_root \
  --decode_dir=textsum/log_root/decode \
  --beam_size=8
```

После ночи вычислений для "игрушечного" примера с рекомендуемыми параметрами ничего не получаем. Выясняем, что как сказано здесь и здесь, модель может тренироваться неделю (CPU/GPU) на выборках > 100к статей и давать плохие результаты.