

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 tutorials 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)

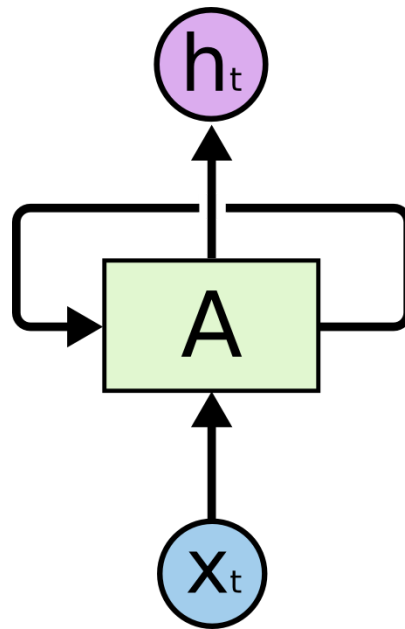
TensorFlow / общая информация

- 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++

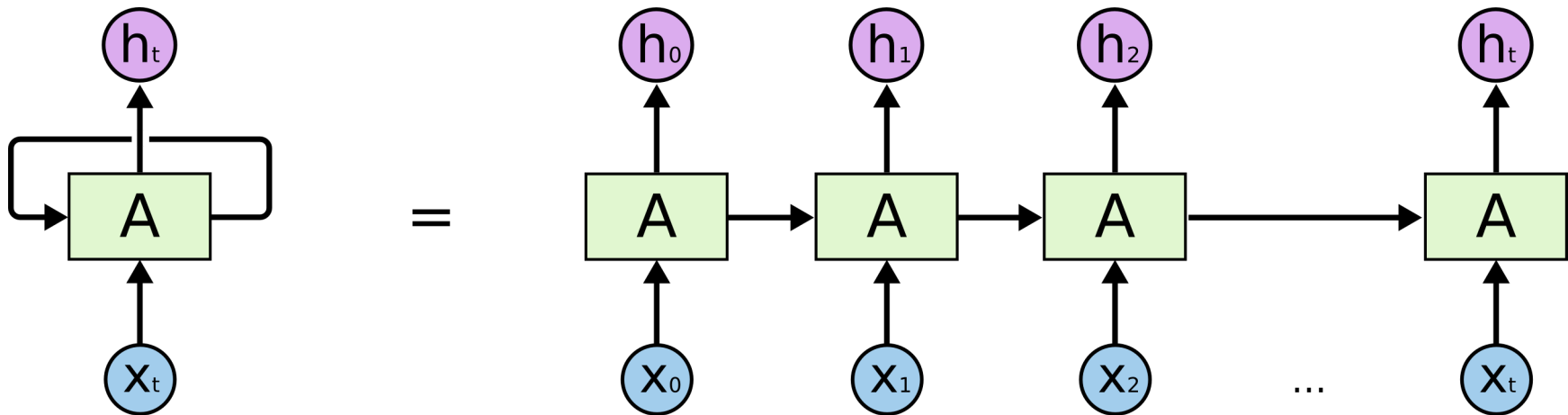
TensorFlow / простой пример

```
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()
```

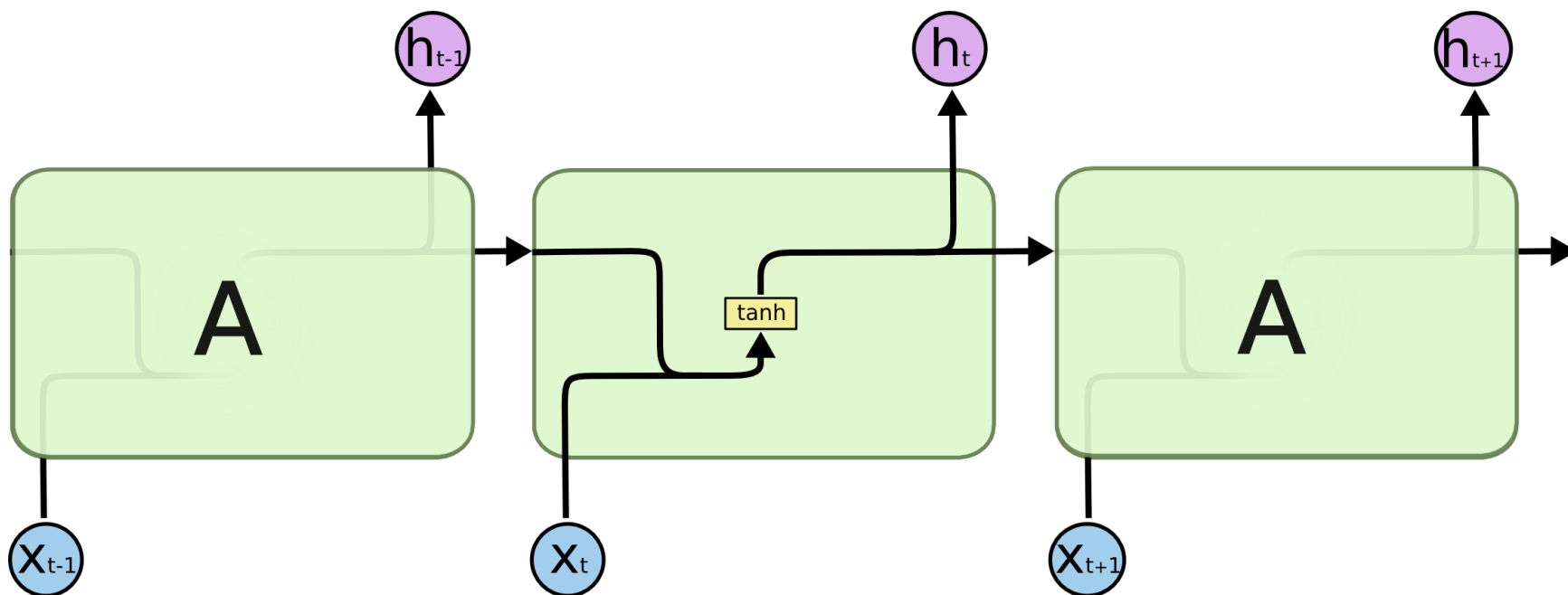
RNN



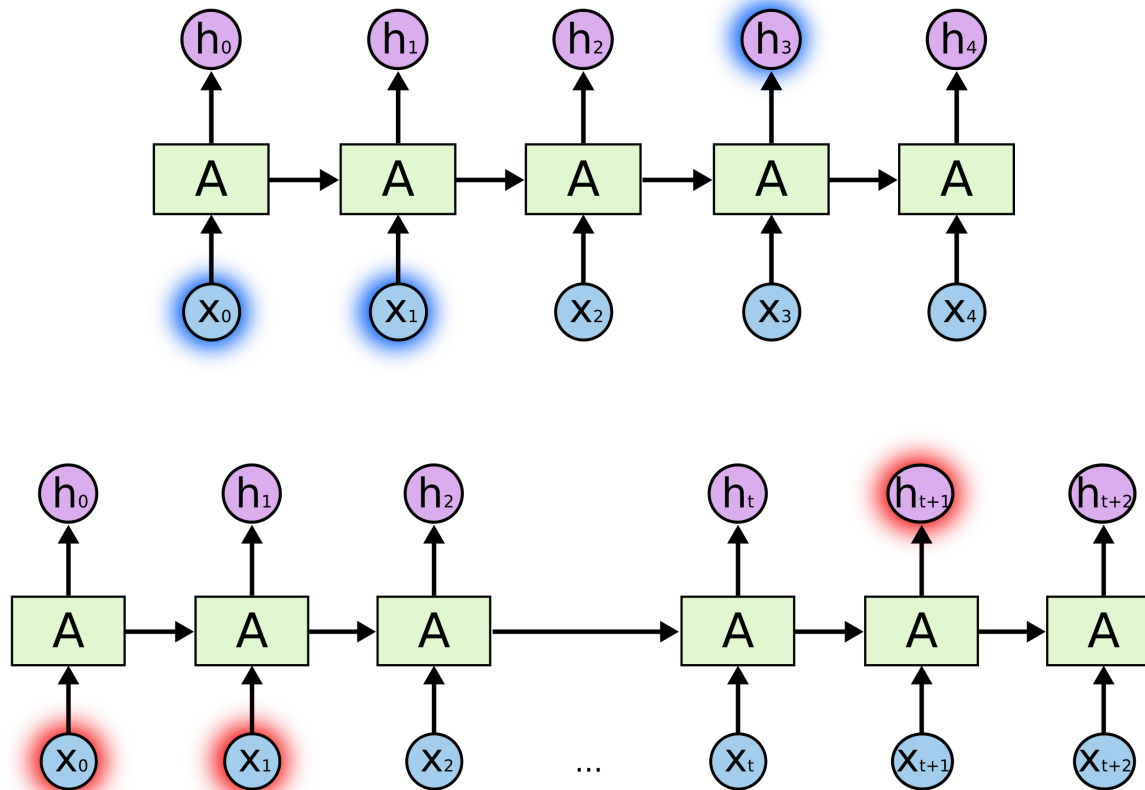
RNN



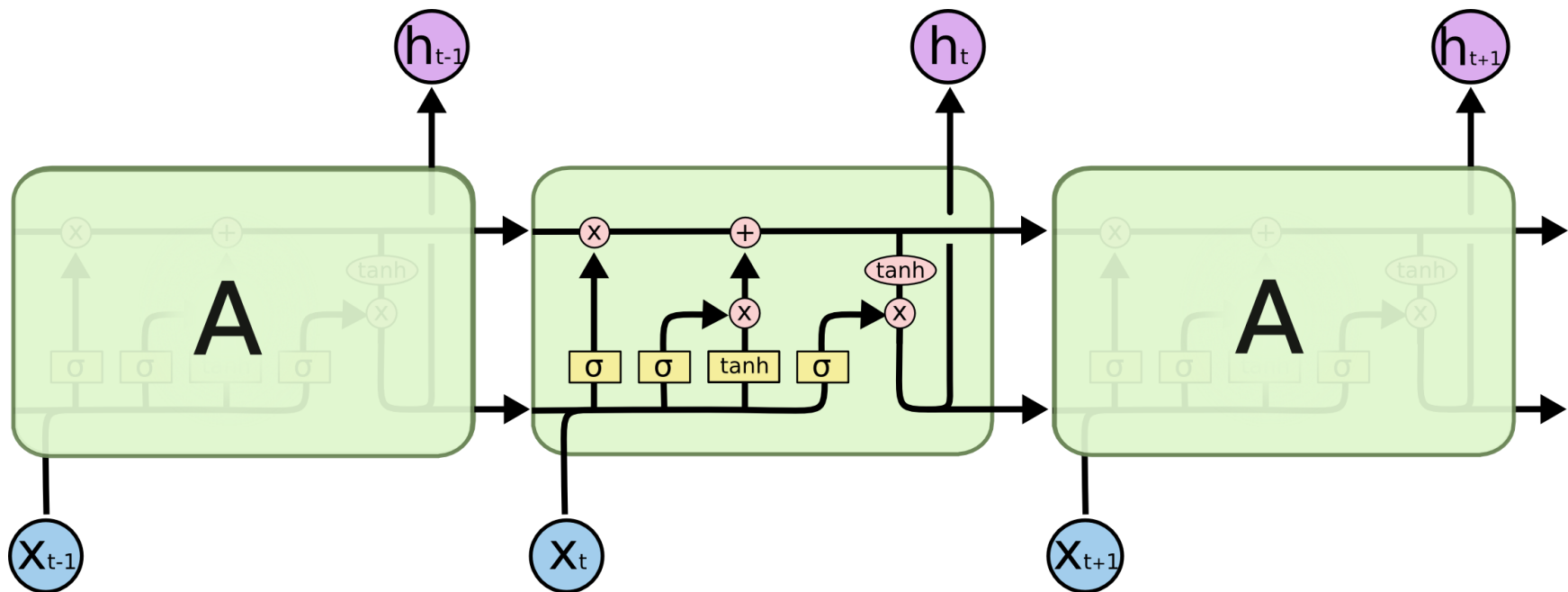
RNN традиционная



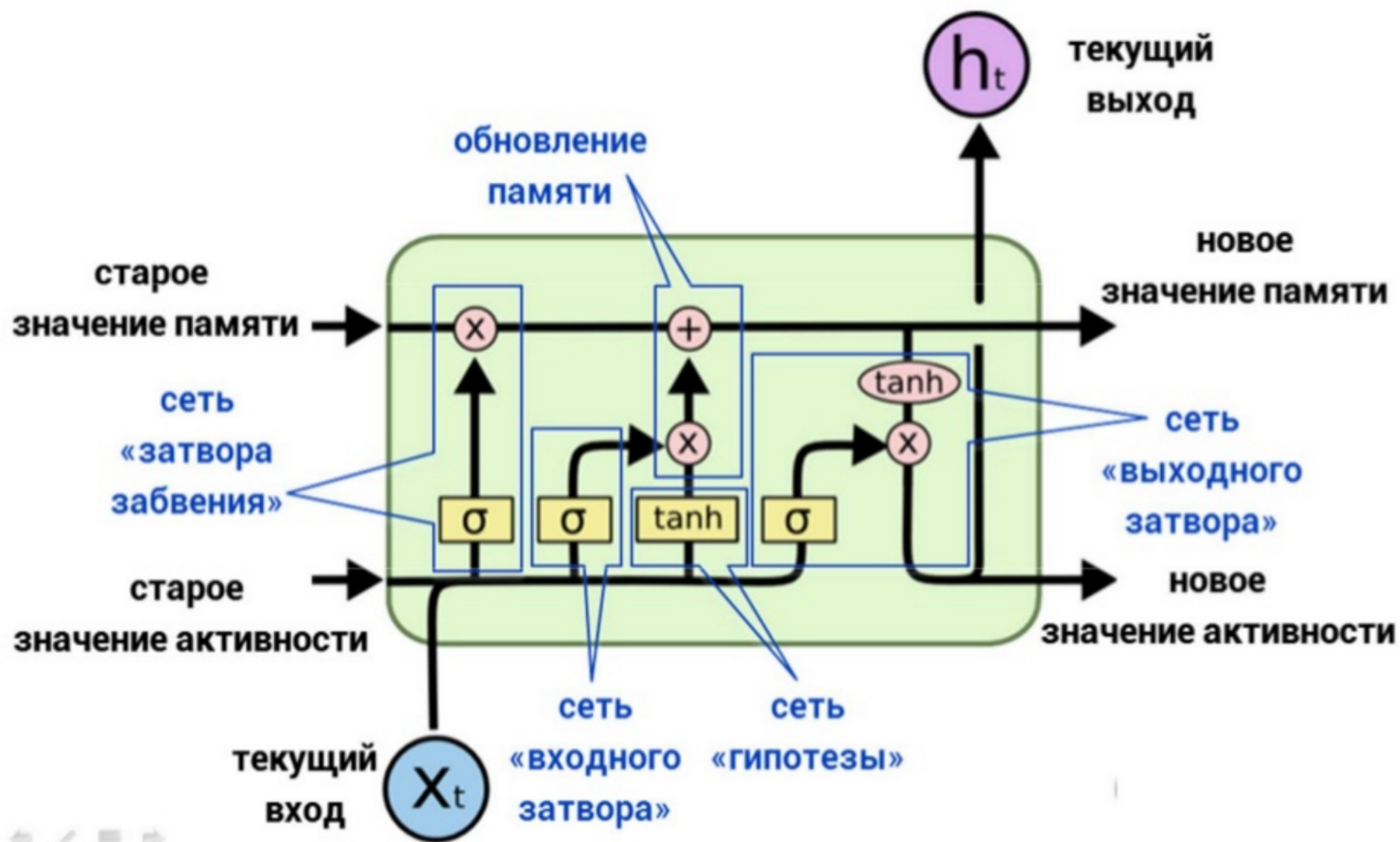
RNN проблема зависимостей



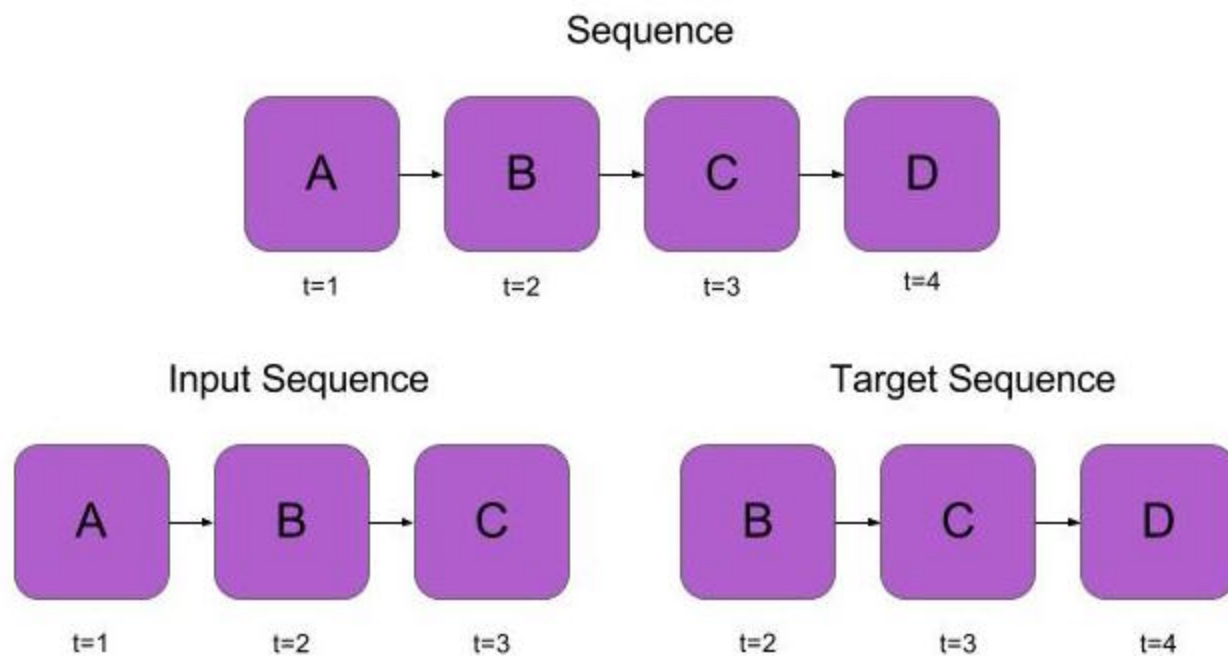
LSTM



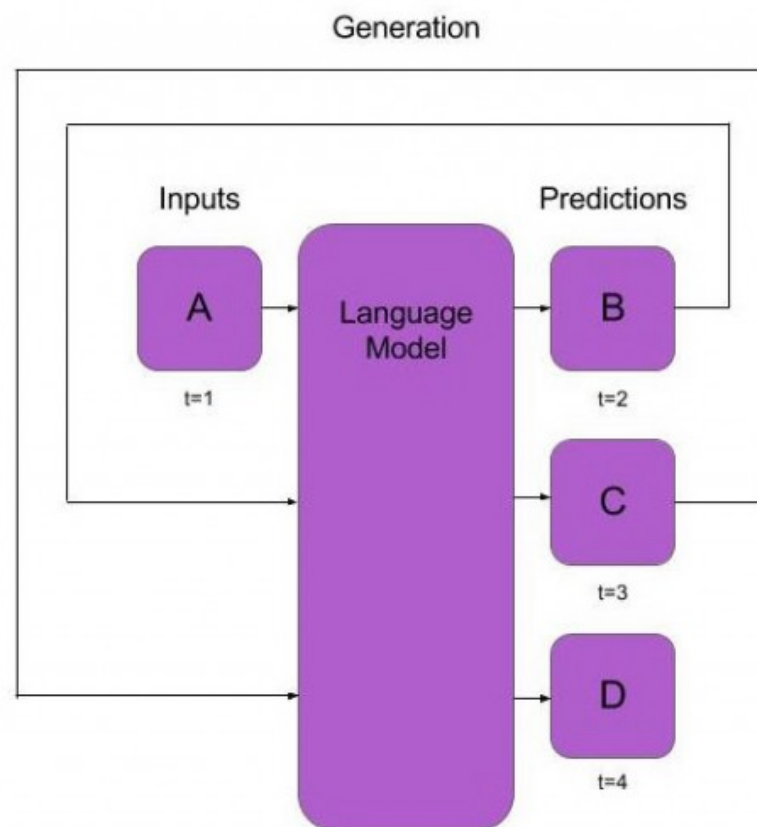
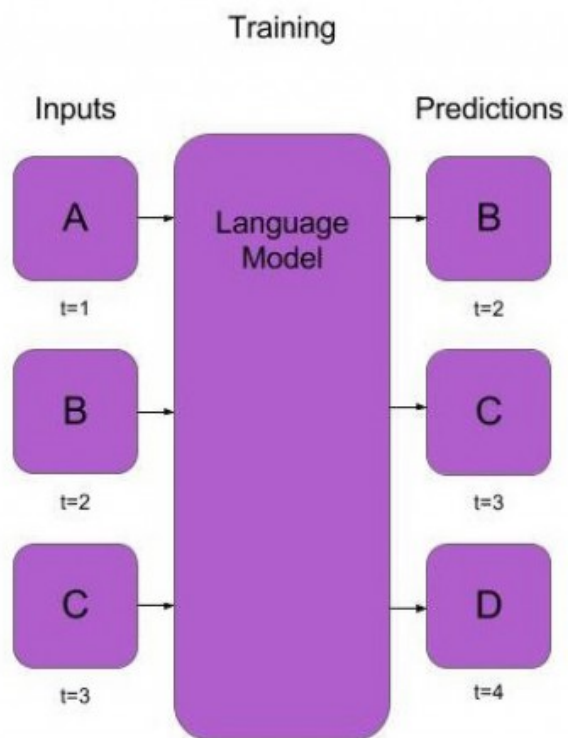
LSTM



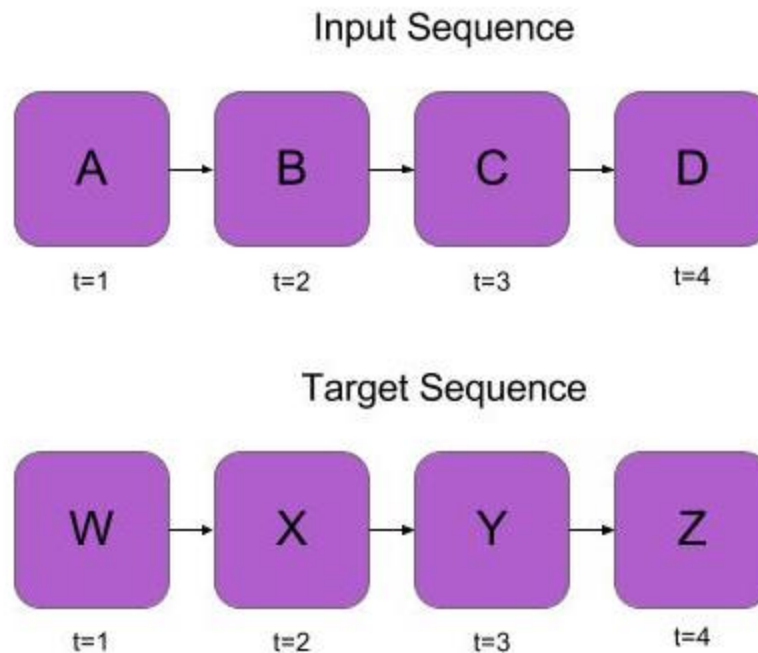
Базовая задача



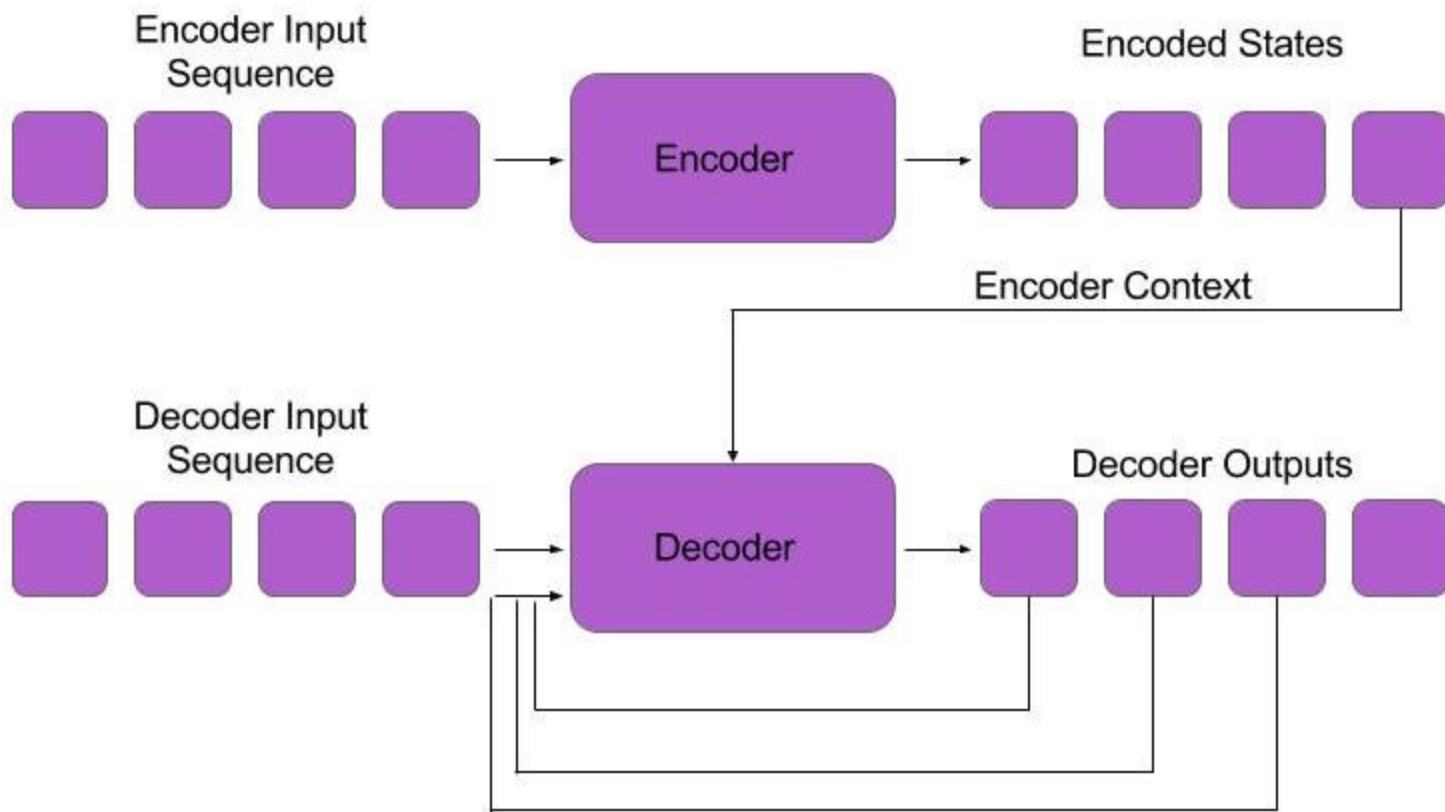
Модель для решения задачи



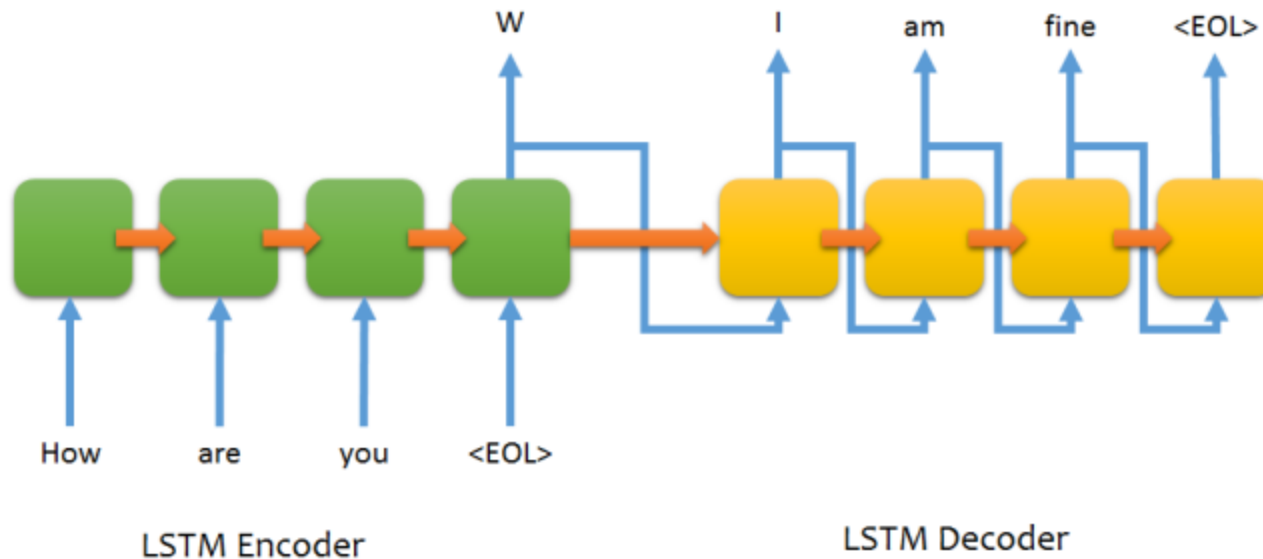
Задача для sequence-to-sequence



Sequence-to-sequence модель

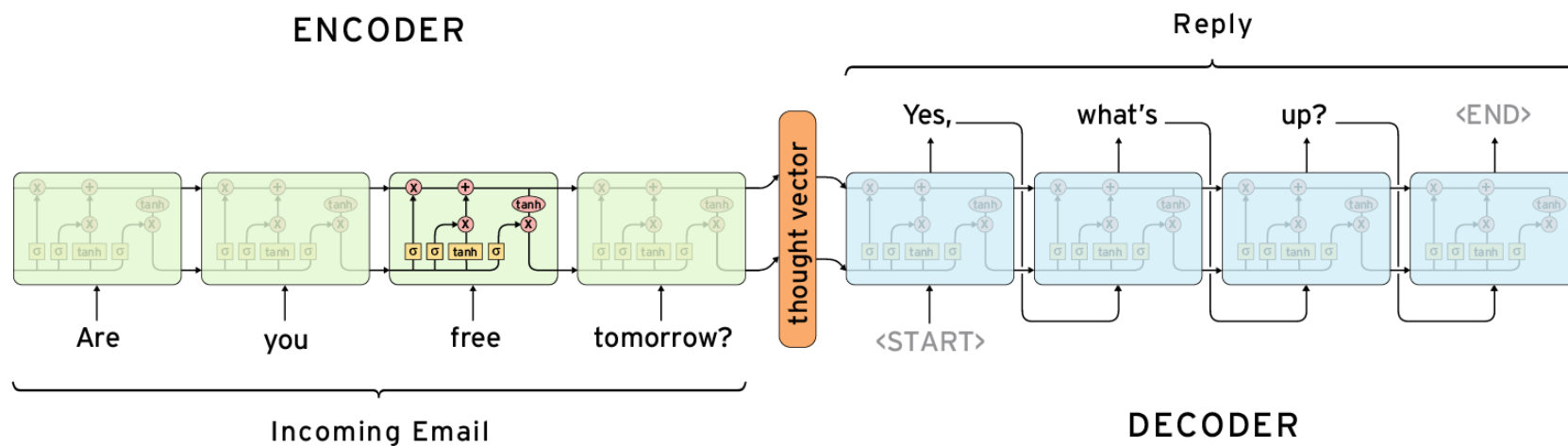


Sequence-to-sequence модель

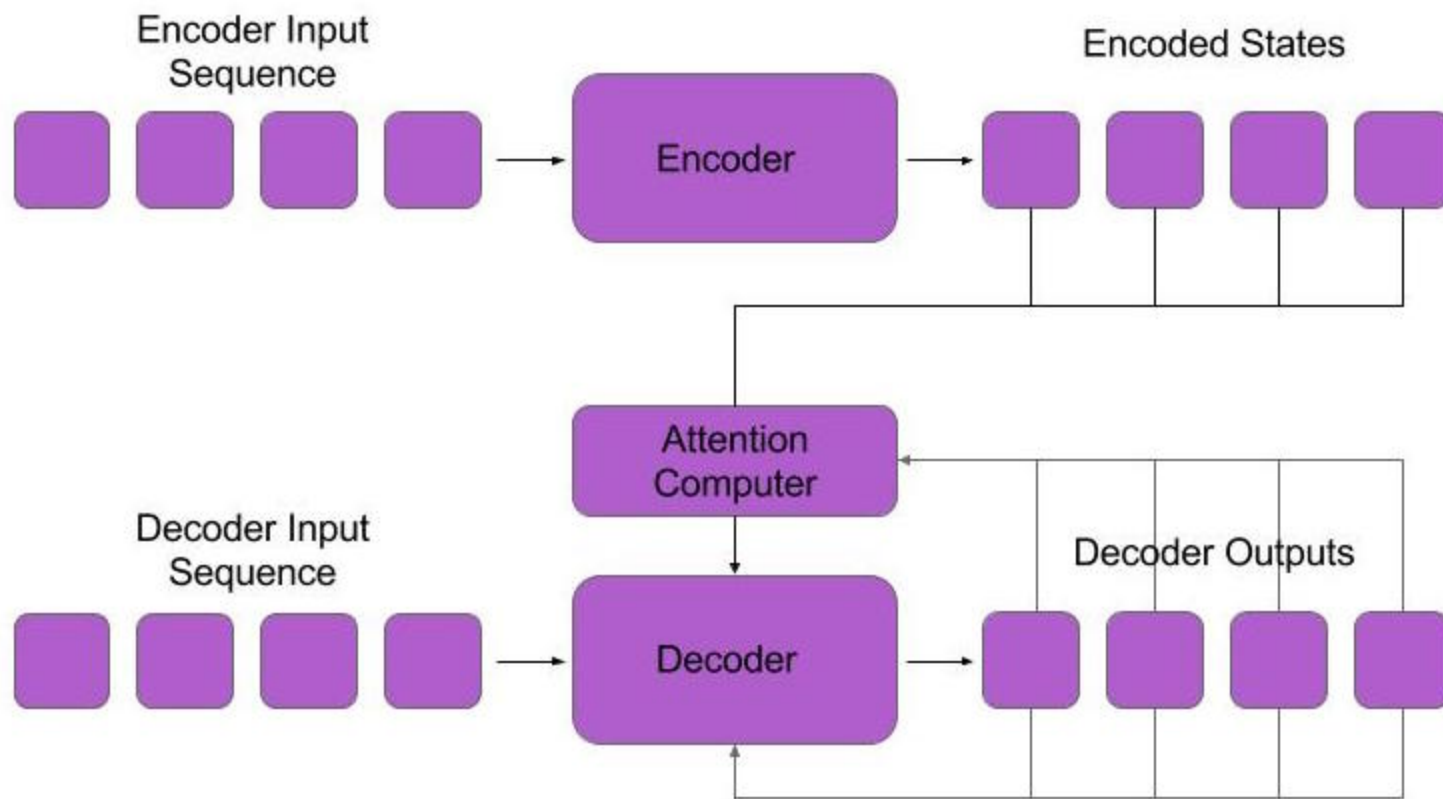


- Каждый прямоугольник - ячейка RNN (GRU или LSTM)
- Encoder и decoder используют различный набор параметров

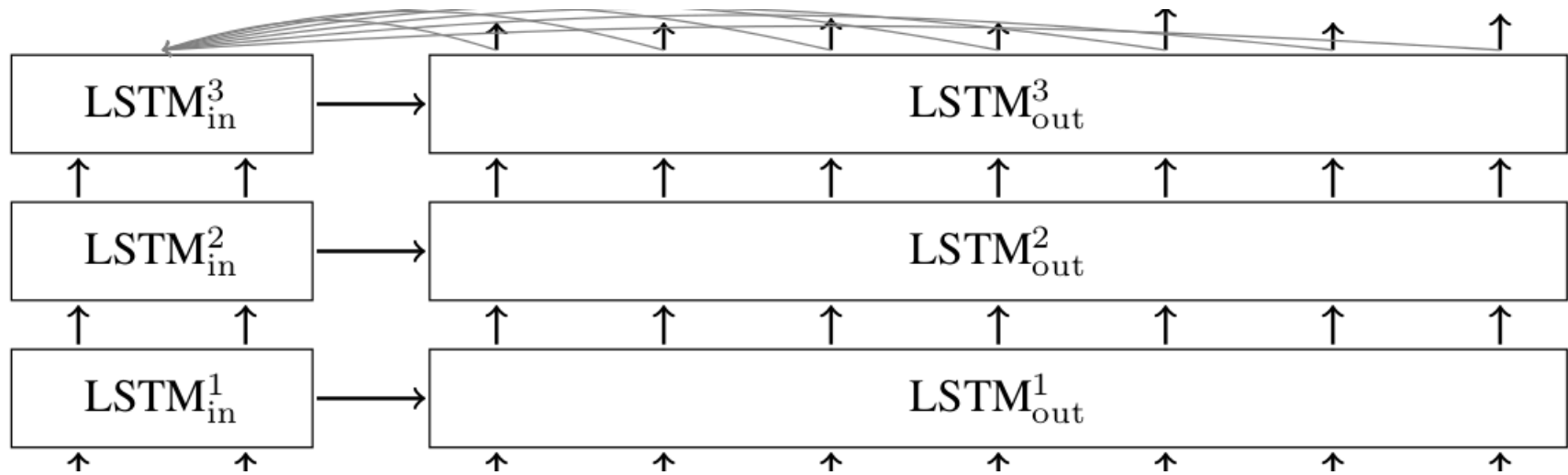
Sequence-to-sequence модель



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 tutorial [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.

Машинный перевод с TensorFlow

Установка

```
git clone https://github.com/tensorflow/tensorflow.git
```

Тренировка модели

оригинальный пример (не работает из коробки):

```
python translate.py --data_dir [your_data_directory]
```

Машинный перевод с TensorFlow

Тестовый пример

- исправления `create_model` (`translate.py`) для запуска тестового примера.
- необходимо удалить префикс "fixed" в автоматически загруженных данных.

Тренировка модели

```
python translate.py --data_dir --data_dir [your_data_directory]  
--train_dir [checkpoints_directory] --size=256 --num_lo
```

Использование модели

```
python translate.py --decode --data_dir [your_data_directory] --  
--size=<size_on_train> --steps_per_checkpoint=50
```

Машинный перевод с TensorFlow

Собственный пример

- Создайте `giga-fren.release2.en` со своими данными в `data_dir`
- Создайте `giga-fren.release2.fr` со своими данными в `data_dir`
- формат данных:
 - текстовые файлы
 - файл исходных последовательностей (англ)/файл последовательностей иного языка(франц)
 - новое предложение на новой строке
 - взаимнооднозначное соответствие последовательностей

Машинный перевод с TensorFlow

Собственный пример/тренировка модели

- size - размер слоя (количество юнитов)
- num_layers - количество слоев в модели
- en_vocab_size/fr_vocab_size - длина словаря
- data_dir - директория с данными
- train_dir - директория для чекпоинтов
- max_train_data_size - объем выборки (количество предложений)
- steps_per_checkpoint - как часто совершать чекпоинт

Машинный перевод с TensorFlow

```
global step 2600 learning rate 0.4707 step-time 6.22 perplexity 83.03
  eval: bucket 0 perplexity 42.16
  eval: bucket 1 perplexity 68.98
  eval: bucket 2 perplexity 87.92
  eval: bucket 3 perplexity 119.44
global step 2650 learning rate 0.4707 step-time 6.96 perplexity 82.04
  eval: bucket 0 perplexity 47.60
  eval: bucket 1 perplexity 72.13
  eval: bucket 2 perplexity 84.98
  eval: bucket 3 perplexity 107.88
global step 2700 learning rate 0.4707 step-time 6.24 perplexity 80.23
  eval: bucket 0 perplexity 117.93
  eval: bucket 1 perplexity 137.24
```

Машинный перевод с TensorFlow

Использование модели

```
python translate.py --decode --data_dir [your_data_directory]  
                    --train_dir [checkpoints_directory] --size=<size_on_train>
```

```
ValueError: ('Vocabulary file %s not found.', 'translate/test/vocab40000.en')  
jule@Alice:~/Documents/IDA/labs/savchenko$ python tensorflow/tensorflow/models/rnn/translate/translate.py  
train_dir translate/train --size=256 --num_layers=2  
Reading model parameters from translate/train/translate.ckpt-2950  
> Who is the President of the United States?  
La Programme du la programme du la programme ?  
> Republican leaders justified their policy by the need to combat electoral fraud.  
Les résultats de la résultats de la programme de la _UNK .  
> Results of the program were obtained  
La programmes de la programmes de la programme  
> Program program program is the best word set for this model  
Le programme du programme de la programme de la programme de la programme de la programme .  
> █
```

