НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Language understanding (CNTK). Применение библиотеки Microsoft CNTK для построения нейросетевых моделей текстов.

Состав команды:

Кузнецов Владимир
Вороная Ксения
Куренков Евгений

Цель нашей работы: Language Understanding with Recurrent Networks

План работы:

- ❑ Первичное изучение библиотеки
https://github.com/Microsoft/CNTK/wiki
https://www.microsoft.com/en-us/research/publication/an-introduction-to-computational-networks-and-the-computational-network-toolkit/
- ❑ Установка и настройка
- ❑ Обучение моделей, запуск и разбор готовых примеров:

ATIS https://github.com/Microsoft/CNTK/tree/master/Examples/Text/ATIS

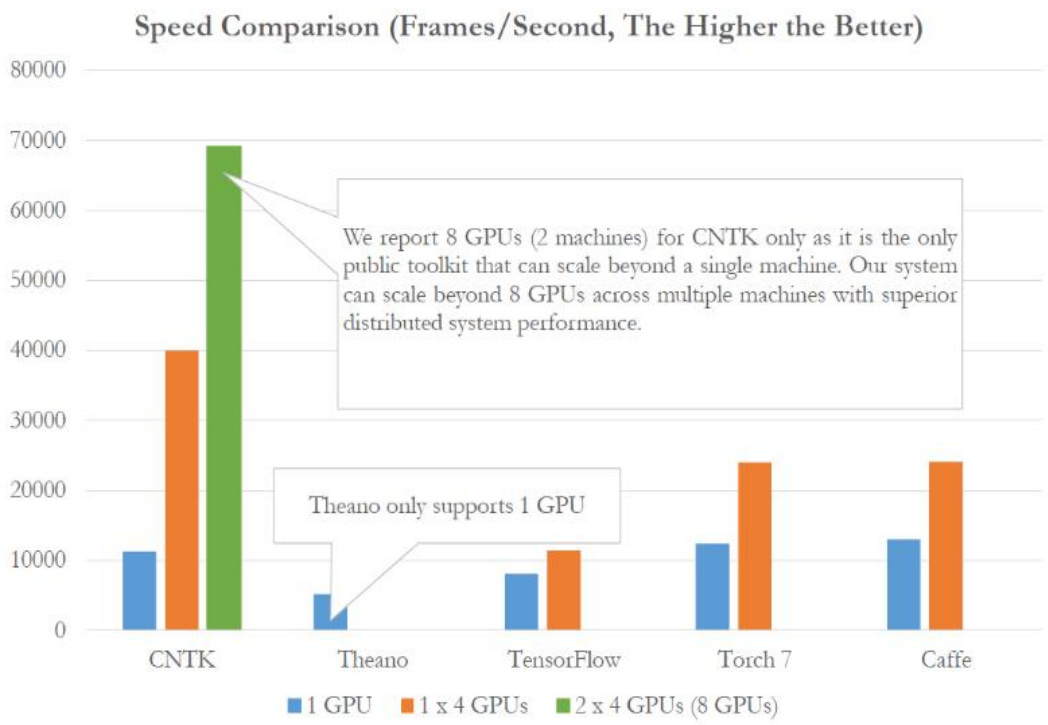
PennTreebank

- ❑ Написание собственной утилиты

Наш репозиторий:
https://github.com/artezio-kseniav/hse_CNTK

Computational Network Toolkit - набор инструментов, созданный командой Microsoft Research, для проектирования и тренировки сетей различного типа, которые можно использовать для распознавания образов, понимания речи, анализа текстов и многого другого.

Сеть от CNTK победила в конкурсе ImageNet LSVR 2015 и является самой быстрой среди существующих конкурентов.



Установка (2 пути):

Через бинарники на Linux 64bit и Windows (были добавлены в августе 2016)

https://github.com/Microsoft/CNTK/releases

для Windows:

- Visual C++ Redistributable Package for Visual Studio 2013
- Microsoft MPI of version 7 (7.0.12437.6)

для Linux

- C++ Compiler
- Open MPI

For GPU systems latest NVIDIA driver.

Через Docker Containers

https://www.docker.com/products/docker-toolbox

Загрузить необходимые докер-файлы здесь
https://github.com/Microsoft/CNTK/tree/master/Tools/docker

Сбилдить контейнер:

```
docker build -t cntk CNTK-CPUOnly-Image
```

где cntk имя нашего образа

Готовый образ можно взять здесь:
https://hub.docker.com/r/torumakabe/cntk-cpu/

```
docker pull torumakabe/cntk-cpu
```

Запустить образ а затем пример:

```
docker run -it --rm cntk
cd Examples/Text/PennTreebank/Data
cntk configFile=../Config/rnn.cntk
```

ATIS example

Цель: создать и обучить рекуррентную нейронную сеть для задач тегирования и классификации данных от Air Travel Information Services (ATIS).
Tasks of slot tagging and intent classification.

Необходимое:
Recurrent neural network
Word embedding (векторное представление слов)

Запуск примера:

```
cd Examples/Text/ATIS
cntk configFile=ATIS.cntk
```

Strongly recommend to run this on a machine with a capable CUDA-compatible GPU. Deep learning without GPUs is not fun.

ATIS example

```
BOS i would like to find a flight from charlotte to Las Vegas that makes a stop in St. Louis EOS
```

it is converted into the following text:

```
1   |PW 1:1     |CW 1:1     |NW 12:1    |L 126:1
1   |PW 1:1     |CW 12:1    |NW 39:1    |L 126:1
1   |PW 12:1    |CW 39:1    |NW 28:1    |L 126:1
1   |PW 39:1    |CW 28:1    |NW 3:1     |L 126:1
1   |PW 28:1    |CW 3:1     |NW 86:1    |L 126:1
1   |PW 3:1     |CW 86:1    |NW 15:1    |L 126:1
1   |PW 86:1    |CW 15:1    |NW 10:1    |L 126:1
1   |PW 15:1    |CW 10:1    |NW 4:1     |L 126:1
1   |PW 10:1    |CW 4:1     |NW 101:1   |L 126:1
1   |PW 4:1     |CW 101:1   |NW 3:1     |L 48:1
1   |PW 101:1   |CW 3:1     |NW 92:1    |L 126:1
1   |PW 3:1     |CW 92:1    |NW 90:1    |L 78:1
1   |PW 92:1    |CW 90:1    |NW 33:1    |L 123:1
1   |PW 90:1    |CW 33:1    |NW 338:1   |L 126:1
1   |PW 33:1    |CW 338:1   |NW 15:1    |L 126:1
1   |PW 338:1   |CW 15:1    |NW 132:1   |L 126:1
1   |PW 15:1    |CW 132:1   |NW 17:1    |L 126:1
1   |PW 132:1   |CW 17:1    |NW 72:1    |L 126:1
1   |PW 17:1    |CW 72:1    |NW 144:1   |L 71:1
1   |PW 72:1    |CW 144:1   |NW 2:1     |L 119:1
1   |PW 144:1   |CW 2:1     |NW 2:1     |L 126:1
```

На вход:
human-computer queries.

Task will be to annotate (tag) each word of a query whether it belongs to a specific item of information (slot), and which one.

Приводим данные к CNTK Text Format.

ATIS example

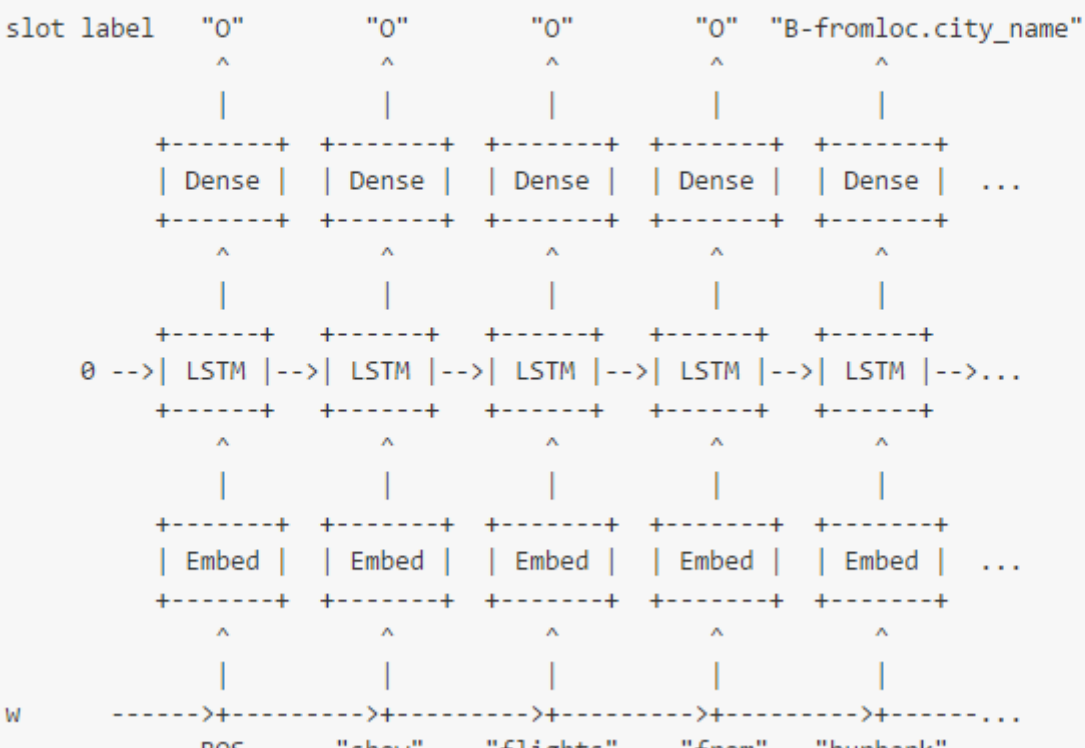
```
19  |S0 178:1 |# BOS      |S1 14:1 |# flight  |S2 128:1 |# O
19  |S0 770:1 |# show                         |S2 128:1 |# O
19  |S0 429:1 |# flights                      |S2 128:1 |# O
19  |S0 444:1 |# from                         |S2 128:1 |# O
19  |S0 272:1 |# burbank                      |S2 48:1  |# B-fromloc.city_name
19  |S0 851:1 |# to                           |S2 128:1 |# O
19  |S0 789:1 |# st.                          |S2 78:1  |# B-toloc.city_name
19  |S0 564:1 |# louis                        |S2 125:1 |# I-toloc.city_name
19  |S0 654:1 |# on                           |S2 128:1 |# O
19  |S0 601:1 |# monday                       |S2 26:1  |# B-depart_date.day_name
19  |S0 179:1 |# EOS                          |S2 128:1 |# O
```

S0 - contains numeric word indices
S1 - is an intent label
S2 - is the slot label, represented as a numeric index

Задача нейронной сети принять запрос (S0) и предсказать slot label (S2).

ATIS example

The model we will use is a recurrent model consisting of an embedding layer, a recurrent LSTM cell, and a dense layer to compute the posterior probabilities:




```
model = Sequential (
    EmbeddingLayer {150} :
    RecurrentLSTMLayer {300} :
    DenseLayer {labelDim}
)
```

ATIS example

Обучение модели происходит в 20 эпох (этапов).

The epoch size is the number of samples--counted as word tokens, not sentences--to process between model checkpoints.

```
C:\testcntk\Examples\Text\ATIS>cntk configFile=ATIS.cntk >> resnew.txt
CNTK 1.7.2 (HEAD d1ad5f, Oct  1 2016 14:16:55) on ksenia-pc at 2016/10/19 03:18:
24

cntk  configFile=ATIS.cntk

################################################################################
#                                                                              #
# Train command (train action)                                                 #
#                                                                              #
################################################################################

Node 'lstmStack.layers[0].lstmState._.ot._.PlusArgs[0].PlusArgs[0].PlusArgs[1].T
imesArgs[0]' (LearnableParameter operation) operation: Tensor shape was inferred
 as [300 x 150].
Node 'lstmStack.layers[0].lstmState._.ft._.PlusArgs[0].PlusArgs[0].PlusArgs[1].T
imesArgs[0]' (LearnableParameter operation) operation: Tensor shape was inferred
 as [300 x 150].
Node 'lstmStack.layers[0].lstmState._.it._.PlusArgs[0].PlusArgs[0].PlusArgs[1].T
imesArgs[0]' (LearnableParameter operation) operation: Tensor shape was inferred
 as [300 x 150].
Node 'lstmStack.layers[0].lstmState._.bit.ElementTimesArgs[1].z.PlusArgs[0].Plus
Args[1].TimesArgs[0]' (LearnableParameter operation) operation: Tensor shape was
 inferred as [300 x 150].

Model has 61 nodes. Using CPU.

Training criterion:   cr = CrossEntropyWithSoftmax
Evaluation criterion: errs = ClassificationError

Training 1005127 parameters in 18 parameter tensors.
```

ATIS example

```
Finished Epoch[ 1 of 20]: [Training] cr = 0.40189162 * 36006; errs = 8.254% * 36
006; totalSamplesSeen = 36006; learningRatePerSample = 0.0099999998; epochTime=3
1.1506s
Finished Epoch[ 2 of 20]: [Training] cr = 0.10937663 * 36001; errs = 2.442% * 36
001; totalSamplesSeen = 72007; learningRatePerSample = 0.0099999998; epochTime=3
0.4011s
Finished Epoch[ 3 of 20]: [Training] cr = 0.05090462 * 36004; errs = 1.189% * 36
004; totalSamplesSeen = 108011; learningRatePerSample = 0.0049999999; epochTime=
32.1833s
Finished Epoch[ 4 of 20]: [Training] cr = 0.04066215 * 35997; errs = 0.958% * 35
997; totalSamplesSeen = 144008; learningRatePerSample = 0.0049999999; epochTime=
32.3319s
```

```
Allocating matrices for forward and/or backward propagation.

Memory Sharing: Out of 61 matrices, 0 are shared as 0, and 61 are not shared.

Minibatch[1-1]: errorRate = 0.02039330 * 10984
Final Results: Minibatch[1-1]: errorRate = 0.02039330 * 10984

Action "test" complete.

COMPLETED.
```

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Спасибо за внимание!