README
datasets
raw_datasets - downloaded data that we use
b0d17d4b33b47a73c915c702b5dfaf81 raw_datasets.zip 603M
processed data are provided separately (including reptil pretrained model):
bf0bf8735e04980ea0173fc634d14ad9 processed_data.zip 339M

dataset preprocessing pipeline:

1. **notebooks/dh_cleaning_datasets_feb_6_more_data.ipynb** which processes datasets/training/validation sets and creates corpus_paragraph_feb_6.txt
2. split long paragraphs into chunks with **cut_into_chunks.py**:

cat corpus_paragraph_feb_6.txt | python cut_into_chunks.py 150 100 > corpus_paragraph_short150-100.txt

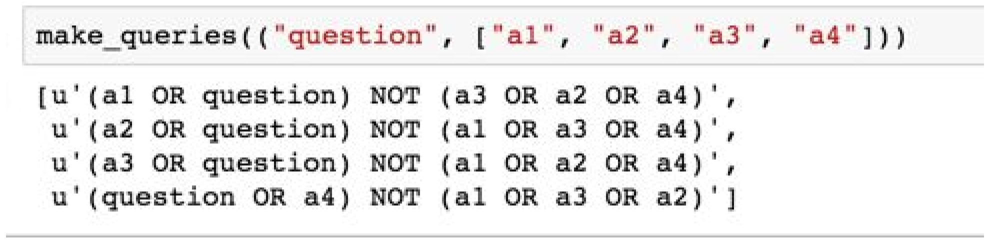
We have several models that provide scores/features, used in the ensemble model as belows:
models

reptil model - aka adaptive skip gram (http://arxiv.org/abs/1502.07257) trained on corpus_paragraph_unstable_short150-100.txt (nicer versions of word2vec)
Notes: reference https://github.com/sbos/AdaGram.jl implementation is used. Python API was implemented by us.
Scores computed here: **notebooks/dh_adam_wrapper.ipynb**
Model is trained like this: sh train.sh --min-freq 20 --window 5 --workers 40 --epochs 5 --dim 300 --alpha 0.1 /root/data/allen-ai-challenge/corpus_paragraph_unstable_short150-100.txt /root/data/allen-ai-challenge/adam.dict /root/data/allen-ai-challenge/adam.model
Training takes about 1hr on 48 cpus
Created adam/model and adam.dict are provided.

n-grams models
First, run make_cooccurences.py for counting ngrams (requires Spark) 4 times, see help.
Then run NGrams_final_scores.ipynb to compute scores used by the ensemble model below.
Training time: counting stats take 1hr (make_cooccurences.py on Spark cluster, 5 machines), can be done on a single machine.

topic modelling

lucene search
lucene indexes are created from corpus_paragraph_short150-100.txt in
dh_cleaning_datasets_feb_6_more_data.ipynb
For each question/answers row we create queries (see the example below), get 30 documents and report their scores to the ensemble model.



```
make_queries(("question", ["a1", "a2", "a3", "a4"]))
```

```
[u'(a1 OR question) NOT (a3 OR a2 OR a4)',
 u'(a2 OR question) NOT (a1 OR a3 OR a4)',
 u'(a3 OR question) NOT (a1 OR a2 OR a4)',
 u'(question OR a4) NOT (a1 OR a3 OR a2)']
```

Creating indices takes around 20 minutes.

ensemble model (notebooks/dh_ensembling.ipynb)
is a calibrating/averaging of the following models:

1. merged_ngrams (scores calculated by n-grams models) ngram
2. reptil features - reptil
3. lucene scores on the whole dataset (lucene_big)
4. lucene scores on the whole dataset minus the latest update (from newdata folder) lucene_small

We calibrate models (all the features from the models are used to minimize cross-entropy in the train set of questions) and run a weighted average on them:
result = 2*lucene_big + lucene_small + reptil + ngram
Model is trained for 5 minutes.

parsers:
wiki parsers: see parse_wiki.py
qiuzlet cards is scraped in dh_scrape_quizlet.ipynb
studystack is downloaded via /topkek_loader code
ck12 parser - Scrapy parser for ck12 data is ck12_parse.py

Hardware requirements:
no GPU :)
we used 48CPU 64GM server for most of the ML training.
Runtime is fast (3 sec per the whole validation set)

Software requirements:
the code is mostly in Pythom with a bit of Julia and Java code
ipython notebooks are run with Jupyter Notebook
(http://jupyter.readthedocs.org/en/latest/install.html)