Speech Recognition

Crow

Crow

Deep Speech import

```
Loading TSV file:  /home/kenneth/Projects/JSALT_NPLM_data/Speech/Deep_Speech/cro/train.tsv
Saving new DeepSpeech-formatted CSV file to:
/home/kenneth/Projects/JSALT_NPLM_data/Speech/Deep_Speech/cro/clips/train.csvImporting mp3
files...Progress
|##################################################################################################
100% completedWriting CSV file for DeepSpeech.py as:
/home/kenneth/Projects/JSALT_NPLM_data/Speech/Deep_Speech/cro/clips/train.csvProgress
|##################################################################################################
100% completed
Imported 50975 samples.
Skipped 278 samples that failed on transcript validation.Skipped 32 samples that were too
short to match the transcript.
Skipped 254 samples that were longer than 10 seconds.Final amount of imported audio: 52:28:24.
Loading TSV file:  /home/kenneth/Projects/JSALT_NPLM_data/Speech/Deep_Speech/cro/test.tsv
Saving new DeepSpeech-formatted CSV file to:
/home/kenneth/Projects/JSALT_NPLM_data/Speech/Deep_Speech/cro/clips/test.csvImporting mp3
files...Progress
|##################################################################################################
|  99% completedWriting CSV file for DeepSpeech.py as:
/home/kenneth/Projects/JSALT_NPLM_data/Speech/Deep_Speech/cro/clips/test.csvProgress
|##################################################################################################
100% completed
Imported 6374 samples.
Skipped 23 samples that failed on transcript validation.Skipped 2 samples that were too short
to match the transcript.
Skipped 31 samples that were longer than 10 seconds.Final amount of imported audio: 6:34:09.
Loading TSV file:  /home/kenneth/Projects/JSALT_NPLM_data/Speech/Deep_Speech/cro/dev.tsv
Saving new DeepSpeech-formatted CSV file to:
/home/kenneth/Projects/JSALT_NPLM_data/Speech/Deep_Speech/cro/clips/dev.csvImporting mp3
files...Progress
|##################################################################################################
|  99% completedWriting CSV file for DeepSpeech.py as:
/home/kenneth/Projects/JSALT_NPLM_data/Speech/Deep_Speech/cro/clips/dev.csvProgress
```

```
|#############################################################################################
100% completed
Imported 6371 samples.
Skipped 36 samples that failed on transcript validation.Skipped 2 samples that were too short
to match the transcript.
Skipped 34 samples that were longer than 10 seconds.
Final amount of imported audio: 6:34:15.
```

Unique characters to add to alphabet

```
['b', 'm', 'f', ' ', 'e', 'r', 'ú', 'l', 'd', '\uf009', 'k', 'c', 'g', 'i', 'a', 'h', 'p',
'u', 'o', 'w', 'á', 't', 's', 'n', 'x']
```

the kenlm model needs to be built. Use the data fran sent and check this out
https://github.com/mozilla/DeepSpeech/issues/1411

Crow res

After setting alpha and beta parameters for language model to 0 (effectively negating the language model)

```
        --lm_weight .1 \
        --lm_alpha 0\
        --drop_source_layers 1 \        --source_model_checkpoint_dir
"${SOURCE_MODEL}/deepspeech-0.5.1-checkpoint/" \
        --n_hidden 2048 \
        --epoch -10 \
        --earlystop_nsteps 5 \
        --train_batch_size 30\
        --dev_batch_size 48 \
        --test_batch_size 48 \
        --learning_rate 0.001 \
        --dropout_rate 0.2 \
```

Test on /home/kenneth/Projects/JSALT_NPLM_data/Speech/Deep_Speech/cro/clips/test.csv - WER: 0.990699, CER: 0.871154, loss: 27.086073

Examples:

```
--------------------------------------------------------------------------------WER:
2.000000, CER: 0.800000, loss: 10.316745
 - src: "grgacgicr " - res: "r r"
--------------------------------------------------------------------------------WER:
1.000000, CER: 0.666667, loss: 1.255609
 - src: "grg" - res: "r"
--------------------------------------------------------------------------------WER:
1.000000, CER: 0.400000, loss: 1.491453
 - src: "bggrp" - res: "bgg"
--------------------------------------------------------------------------------WER:
1.000000, CER: 1.000000, loss: 1.747670
```

```
 - src: "cg" - res: ""
--------------------------------------------------------------------------------WER:
1.000000, CER: 1.000000, loss: 2.060837
 - src: "gahg" - res: ""
--------------------------------------------------------------------------------WER:
1.000000, CER: 1.000000, loss: 2.078961
 - src: "cg" - res: ""
--------------------------------------------------------------------------------WER:
1.000000, CER: 0.500000, loss: 2.292280
 - src: "gg" - res: "bgg"
--------------------------------------------------------------------------------WER:
1.000000, CER: 1.000000, loss: 2.531346
 - src: "cg" - res: "r"
--------------------------------------------------------------------------------WER:
1.000000, CER: 1.000000, loss: 2.713193
 - src: "ccg" - res: ""
--------------------------------------------------------------------------------WER:
1.000000, CER: 1.000000, loss: 2.719913
 - src: "gg" - res: ""
--------------------------------------------------------------------------------
```

```
        --lm_weight .1 \
        --lm_alpha 0\
        --drop_source_layers 1 \        --source_model_checkpoint_dir
"${SOURCE_MODEL}/deepspeech-0.5.1-checkpoint/" \
        --n_hidden 2048 \
        --epoch -10 \
        --earlystop_nsteps 5 \
        --train_batch_size 30\
        --dev_batch_size 48 \
        --test_batch_size 48 \
        --learning_rate 0.0001 \
        --dropout_rate 0.2 \
```

Test on /home/kenneth/Projects/JSALT_NPLM_data/Speech/Deep_Speech/cro/clips/test.csv - WER: 0.989741, CER: 0.857567, loss: 25.778952

```
--------------------------------------------------------------------------------WER:
```

```
2.000000, CER: 0.750000, loss: 7.365357
 - src: "grgágr  " - res: "r r"
--------------------------------------------------------------------------------WER:
1.000000, CER: 0.666667, loss: 0.978152
 - src: "grg" - res: "r "
--------------------------------------------------------------------------------WER:
1.000000, CER: 0.666667, loss: 1.270408
 - src: "grg" - res: "r "
--------------------------------------------------------------------------------WER:
1.000000, CER: 1.000000, loss: 1.365655
 - src: "gsa " - res: "b"
--------------------------------------------------------------------------------WER:
1.000000, CER: 0.750000, loss: 1.641714
 - src: "gahg" - res: "bg"
--------------------------------------------------------------------------------WER:
1.000000, CER: 0.400000, loss: 1.725150
 - src: "bgga " - res: "bgg"
--------------------------------------------------------------------------------WER:
1.000000, CER: 1.000000, loss: 1.827598
 - src: "ggl " - res: "b"
--------------------------------------------------------------------------------WER:
1.000000, CER: 1.000000, loss: 1.866848
 - src: "gacw " - res: "b"
--------------------------------------------------------------------------------WER:
1.000000, CER: 0.714286, loss: 1.869485
 - src: "gáicdi " - res: "di"
--------------------------------------------------------------------------------WER:
1.000000, CER: 0.400000, loss: 1.982887
 - src: "bggrp" - res: "bgg"
--------------------------------------------------------------------------------
```

Text conversion command

```
for file in `ls ../Inuit-
Yupik/ess/monolingual_corpus/sivuqam_nangaghnegha/sivuqam_volume2/volume2.gold.ess/`
do     iconv -f ISO-8859-1 -t UTF-8//TRANSLIT "../Inuit-
Yupik/ess/monolingual_corpus/sivuqam_nangaghnegha/sivuqam_volume2/volume2.gold.ess/$file" |
sponge "../Inuit-
Yupik/ess/monolingual_corpus/sivuqam_nangaghnegha/sivuqam_volume2/volume2.gold.ess/$file"
done
```

Deep speech build

These instructions are for those of us who wish to have a version of Mozilla's DeepSpeech where they can make changes to the `native_client` decoder.

Getting the build environment set up to enable this is a non-trivial process, and we hope that these instructions will guide the reader through the setup in as painless a way as possible.

Many modern practices in software engineering are used, some of which may appear counterintuitive. However, it is possible to use Bazel to build the software, so do not give up when you face adversity!

Set up a virtualenv for everything

Let's say you're in your home directory, make a new directory for all of the source code stuff:

```
$ mkdir Mozilla
$ cd Mozilla
$ virtualenv -p python3.7 tmp/deepspeech-venv/$ source
/home/fran/source/Mozilla/tmp/deepspeech-venv/bin/activate
```

Get the source code

Now check out your fork of DeepSpeech (We presume you have forked it and are intending to submit pull requests)

```
$ git clone git@github.com:ftyers/DeepSpeech.git
```

Replace `ftyers` here with your GitHub username.

Next check out Mozilla's `tensorflow` fork:

```
$ git clone https://github.com/mozilla/tensorflow.git
$ cd tensorflow
$ git checkout origin/r1.14
$ cd ..
```

Next check out `kenlm`:

```
$ git clone git@github.com:kpu/kenlm.git
```

Now run `ls`, you should have the following directories:

```
$ ls
DeepSpeech  kenlm  tensorflow  tmp
```

Install Bazel

In order build the `native_client` stuff, you need Google's `Bazel` build system, each version of tensorflow makes its own requirements on the version of Bazel you need. For tensorflow 1.14, you need Bazel 0.24.1, you can install it using the following commands:

```
$ wget https://github.com/bazelbuild/bazel/releases/download/0.24.1/bazel-0.24.1-installer-linux-x86_64.sh
$ bash bazel-0.24.1-installer-linux-x86_64.sh --prefix=/home/fran/local/bin
```

Note that this shell script installs a binary of `Bazel` that is embedded in a zip file inside the shell script, the --prefix give the location to install it. It should be in your `$PATH`.

Check that Bazel is working by running:

```
$ bazel version
```

Set up necessary symlinks and build options

```
$ cd tensorflow
$ ln -s ../DeepSpeech/native_client/ .
$ ./configure
```

This is an interactive `configure` script. Make sure you are using `python3.7` if you aren't you can set the environment variable `PYTHON_BIN_PATH`.

You definitely want to change the "desired Python library path to use", make sure you change it to something like:

```
/usr/lib/python3.7/dist-packages/
```

If you are building on a laptop without a GPU you can basically say no to all the other questions. You should get something like:

```
$ ./configure WARNING: --batch mode is deprecated. Please instead explicitly shut down your
Bazel server using the command "bazel shutdown".
You have bazel 0.24.1 installed.
Found possible Python library paths:
  /home/fran/local/lib/python3.7/site-packages/
  /usr/lib/python3/dist-packages
  /usr/local/lib/python3.7/dist-packagesPlease input the desired Python library path to use.
Default is [/home/fran/local/lib/python3.7/site-packages/]
Do you wish to build TensorFlow with XLA JIT support? [Y/n]: nNo XLA JIT support will be
enabled for TensorFlow.
Do you wish to build TensorFlow with OpenCL SYCL support? [y/N]: nNo OpenCL SYCL support will
be enabled for TensorFlow.
Do you wish to build TensorFlow with ROCm support? [y/N]: nNo ROCm support will be enabled
for TensorFlow.
Do you wish to build TensorFlow with CUDA support? [y/N]: nNo CUDA support will be enabled
for TensorFlow.
```

```
Do you wish to download a fresh release of clang? (Experimental) [y/N]: nClang will not be downloaded.
Do you wish to build TensorFlow with MPI support? [y/N]: nNo MPI support will be enabled for TensorFlow.
Please specify optimization flags to use during compilation when bazel option "--config=opt" is specified [Default is -march=native -Wno-sign-compare]:
Would you like to interactively configure ./WORKSPACE for Android builds? [y/N]: nNot configuring the WORKSPACE for Android builds.
```

Build `native_client` in the tensorflow directory

```
$ bazel build --workspace_status_command="bash native_client/bazel_workspace_status_cmd.sh" --config=monolithic -c opt --copt=-O3 --copt="-D_GLIBCXX_USE_CXX11_ABI=0" --copt=-fvisibility=hidden //native_client:libdeepspeech.so //native_client:generate_trie
```

If you run out of memory during the build process, specify the number of jobs by adding `--jobs` at the end of the bazel build command, e.g.

```
$ bazel build --workspace_status_command="bash native_client/bazel_workspace_status_cmd.sh" --config=monolithic -c opt --copt=-O3 --copt="-D_GLIBCXX_USE_CXX11_ABI=0" --copt=-fvisibility=hidden //native_client:libdeepspeech.so //native_client:generate_trie --jobs 6
```

Compile the `deepspeech` binary and language bindings

First compile the `deepspeech` binary

```
$ cd ../DeepSpeech/native_client/
$ TFDIR=../../tensorflow make deepspeech
```

Then compile and install the Python bindings

```
$ cd python
$ make bindings
$ pip install dist/deepspeech*
```

And the `ctcdecode` package:

```
$ cd ../ctcdecode/
$ make bindings
$ pip install dist/*.whl
```

Compile kenlm

```
$ cd ../../../kenlm/
$ mkdir build
$ cd build
$ cmake ../
$ make
```

Make a change in the native_client and rebuild it

Make a change in the `native_client` code:

```
$ cd ../../tensorflow/
```

Now edit `native_client/ctcdecode/scorer.cpp`, you can add something like:

```
  std::cerr << "Scorer::setup()" << lm_path << " ||| " trie_path << std::endl;
```

And then try and rebuild:

First you need to run `make clean` in `native_client/ctcdecode`

```
cd native_client/ctcdecode
make clean
cd ../../
```

This is because the build leaves behind some `.h` files that are picked up by a build rule somewhere.

And then you can rebuild:

```
bazel build --workspace_status_command="bash native_client/bazel_workspace_status_cmd.sh" --
config=monolithic -c opt --copt=-O3 --copt="-D_GLIBCXX_USE_CXX11_ABI=0" --copt=-
fvisibility=hidden //native_client:libdeepspeech.so //native_client:generate_trie
```

Now set up some data and try it out

First of all, set up a language model with KenLM:

You'll need some English text, you can get it from OPUS

```
$ cd ../DeepSpeech/data/lm$ wget http://opus.nlpl.eu/download.php?f=SETIMES/v2/moses/en-
tr.txt.zip -O en-tr.txt.zip
$ unzip en-tr.txt.zip
```

```
$ ../../../kenlm/build/bin/lmplz -o 5 < SETIMES.en-tr.en > lm.arpa$
../../../kenlm/build/bin/build_binary lm.arpa lm.binary$ cat SETIMES.en-tr.en | sed
's/./&\n/g' | sort -u > alphabet.txt$ ../../../tensorflow/bazel-
bin/native_client/generate_trie alphabet.txt lm.binary trie
```

Now you can run the demo:

```
$ cd ../../
$ pip3 install -r requirements.txt
$ ./bin/run-ldc93s1.sh
```

It should finish with:

```
I0717 14:56:33.608065 139972947781440 saver.py:1280] Restoring parameters from
/home/fran/.local/share/deepspeech/ldc93s1/train-800I Restored variables from most recent
checkpoint at /home/fran/.local/share/deepspeech/ldc93s1/train-800, step 800Testing model on
data/ldc93s1/ldc93s1.csv
I Test epoch...Test on data/ldc93s1/ldc93s1.csv - WER: 1.000000, CER: 0.980769, loss: 0.005800
--------------------------------------------------------------------------------WER:
1.000000, CER: 0.980769, loss: 0.005800 - src: "she had your dark suit in greasy wash water
all year" - res: "i"
--------------------------------------------------------------------------------
```

The change-build-test loop

If you make a change to `native_client`, you need to go back to the tensorflow directory and run:

```
$ bazel build --workspace_status_command="bash native_client/bazel_workspace_status_cmd.sh" --
config=monolithic -c opt --copt=-O3 --copt="-D_GLIBCXX_USE_CXX11_ABI=0" --copt=-
fvisibility=hidden //native_client:libdeepspeech.so //native_client:generate_trie --
verbose_failures
```

Then after that you need to go back to the DeepSpeech/native_client directory

```
$ cp ../../tensorflow/bazel-bin/native_client/libdeepspeech.so /home/fran/local/lib/
```

for me this is inside my virtualenv e.g.
`~/deep-speech-env/lib/python3.6/site-packages/deepspeech/lib/libdeepspeech.so`

```
2211  cd ../DeepSpeech/native_client/
2212  cd ctcdecode/
2213  make bindings
2216  pip install -U dist/*.whl
2220  cd native_client/
2221  cd python/
2223  make bindings
2226  pip3 install -U dist/*
```

when doing this, something else is missing. e.g. something is hanging around. The rebuilt version is still printing a ton of debug statements even though I have removed these.

Deep Speech decoder breakdown

Generate trie

This issues a call to scorer::setup()?

Deep Speech Scorer Refactor

I have successfully renamed all variables dealing with the scorer and reran everything. The next step is to refactor into a generic and an inherited version.

Potentially using tensorflow api in python instead of c++

the tensorflow api in c++ is a bear to use. no one really uses it much and it has some noteworthy differences from the ubiquitous python api.

the issue is that the scorer in ds_ctcdecoder is built using a bunch of c++ code with swig wrappers to expose the functionality to python.

much of my previous work has involved writing a new scorer in c++ using the c++ tensorflow api and then wrapping this in swig methods like the existing scorer.

i've spent countless hours just trying to get bazel to build things correctly and I have had very limited success.

in particular the error i'm running into now is that I don't know how to build against the flatbuffer library that is required for reading in tensorflow lite model files.

one alternative course of action is to use the python api for tensorflow which comes with the benefit of having more documentation and community support behind it.

the issue is that it's unclear how to integrate python in with the wrapped c++ code in a way that's maintainable for the future of deepspeech.

I cannot modify the swigwrapper.py file as this is autogenerated and is removed upon each make clean in the ctc_decoder directory.

Building mozilla with deepspeech integration

```
https://treeherder.mozilla.org/#/jobs?repo=try&revision=5266680e9e43df3dd2a922b334ce9b33df73aba
Treeherder
you need to pick your build
then click the "task" link in the bottom leftthen in the new page, click on tha "artifatcs" tab
then pick target.zip or target.dmg
that's your firefox build
extract, run with "./firefox -no-remote -profileManager"create a profile named "webspeechapi"
close
run with "./firefox -no-remote -P webspeechapi"once opened, go to about:config, search for webspeech.recognition, toggle both prefs to true
then go to about:deepspeechclick "install lib", wait for the download & extraction to complete
you should then get a reload, and be able to click "install en-us"wait again for download & extraction, it should reload another time, and you should see some model parameters as well as "English (US)"
then you can go to, for example, translate.google.comyou will have a mic in the bottom left of the input box once you select "English" as an input language
please keep that to you
this is **very** preliminary code
I don't have to have people playing with it
at least not outside of my knowledge
```

Datasets

Guarani BABEL is stored at `/nas/data/lorelei/ldc_downloads/LDC2016E19__IARPA_Babel_Guarani_Language_Pack__IARPA-babel305b-v1.0` on qivalluk and kulusiq.

Guarani

Using pytorch deepspeech

Data stats

Conversational training: 37.55 hours

```
find wav/ | xargs -I {} -r soxi -D {} | awk '{sum += $1} END {print sum}'
```

Scripted training: 12.82 hours (using same command in the scripted training dir)

Conversion of sph files

```
(metalearn) (base) kenneth@mwanafunzi:/home/data/corpora/speech/IARPA-babel305b-v1.0c-build$ find -
```

New Page

Early stuff I was doing was seeing the first epochs perform the best and high sensitivity to number of rnn layers (less was performing better) this ended up being because the gradients were exploding.

To solve this, I specified --max-norm 1 which forces gradient clipping.

Ended up getting 66 CER after 10 epochs.

Trying again with 100 epochs and data augmentation (pitch peturbation and gain modulation).

```
 python train.py --train-manifest data/train_combined.csv --val-manifest
data/_home_data_corpora_speech_IARPA-babel305b-v1.0c-
build_converted_BABEL_OP3_305_conversational_dev__manifest.csv --model-path
models/100epoch2.pth --max-norm 1 --epochs 100 --opt-level O0 --cuda --augment --labels-path
data/BABEL/guarani_labels.json --tensorboard
```

```
 python train.py --train-manifest data/train_combined.csv --val-manifest
data/_home_data_corpora_speech_IARPA-babel305b-v1.0c-
build_converted_BABEL_OP3_305_conversational_dev__manifest.csv --model-path
models/100epoch2.pth --max-norm 50 --epochs 100 --opt-level O0 --cuda --augment --labels-path
data/BABEL/guarani_labels.json --tensorboard --augment
```

1 gru successful

```
python train.py --train-manifest data/train_combined.csv --val-manifest
data/_home_data_corpora_speech_IARPA-babel305b-v1.0c-
build_converted_BABEL_OP3_305_conversational_dev__manifest.csv --model-path
models/1_hi_aug.pth --max-norm 100 --hidden-layers 1 --epochs 20 --opt-level O0 --cuda --
labels-path data/BABEL/guarani_labels.json --loss-scale 1 --tensorboard --id
```

```
1_hid_max_norm_100
```

result: WER 88.366 CER 43.116

still achieving better performance after 20 epochs, so I should rerun with more epochs

2 gru

```
python train.py --train-manifest data/train_combined.csv --val-manifest
data/_home_data_corpora_speech_IARPA-babel305b-v1.0c-
build_converted_BABEL_OP3_305_conversational_dev__manifest.csv --model-path
models/2_hi_aug.pth --max-norm 100 --hidden-layers 2 --epochs 40 --opt-level O0 --cuda --
labels-path data/BABEL/guarani_labels.json --loss-scale 1 --tensorboard --id
2_hid_max_norm_100
```

results with max norm = 100

Average WER = 74.590 Average CER = 29.756

Going to try again with higher max norm (default 400) and see how that goes. Before the gradients were exploding when I did that.

3 gru

max norm = 100, after 40 epochs: WER = 60.519 CER = 21.876

4 gru

max norm = 100, after 40 epochs: WER = 52.413 CER = 17.833

5 gru

(ran on qivalluk). 40 epochs, Average WER 14.239 Average CER 3.655. Used nearly all default arguments.