AMD optimized crfsuite

Problem

The bundled binary from python-crfsuite and sklearn-crfsuite performs rather badly on amd processors.

For example, in a trial run with training a small pos tagger on an AMD Epyc 7601, each iteration in the hyperparameter search took about 1 minute 15 seconds. This is only with a small training set of about 400 sentences. With the full 6,000 sentences available it takes upwards of 4 days to finish a single run. Rough.

However, on a dual intel E5-2680 system (16 cores at 3.2 Ghz all core turbo), the performance is much faster. On that same small dataset of 400 sentences, each iteration takes about 30 seconds, the entire hyperparameter search over 50 combinations takes 2 minutes.

This appears to be due to the fact that the binaries that ship with sklearn-crfsuite were compiled on an intel platform.

Solution

To fix this, I created a fork of python-crfsuite that uses the avx2 instructions available on amd's zen processors (this should also help with newer intel processors that have avx extensions).

This fork is available at https://github.com/ksteimel/python-crfsuite.git

To use this, clone the repo

```
git clone --recurse-submodules https://github.com/ksteimel/python-crfsuite.git
```

Then change into the new directory.

```
cd python-crfsuite
```

it's a good idea to create a virtual environment so if you decide you don't want to use this version,

you don't have to.

```
virtualenv -p <your python location> <path to virtualenv>source <path to
virtualenv>/bin/activate
```

Then, we need to build the package.

```
python setup.py build
python setup.py install
```

Now you should have an optimized build of python-crfsuite You can then install sklearn-crfsuite .

```
pip install sklearn-crfsuite
pip install scikit-learn
```

To get out of your virtual environment run,

```
deactivate
```

How did we do?

The whole point of this was to speed up performance on amd processors with crfsuite. If we go back to our small training dataset, we see a substantial boost in performance.

We've now gone from one minute 15 seconds per iteration to only 30ish seconds per iteration.

It may seem somewhat shocking that the amd processor is only about as fast per iteration as the intel processor. However, the amd processor has double the number of cores (32 instead of 16) with a lower clock speed. (2.2 Ghz for the epyc processor compared to 3.2 for the intel procesor). If we look at the entire grid search across 50 parameters, we see the core advantage of the epyc processor emerge: the grid search finished in only 1.1 minutes on the AMD processor while it took 2.0 minutes on the pair of intel processors.

Not too shabby for 2 minutes of work.

Revision #3
Created Thu, Nov 14, 2019 2:45 AM by kenneth
Updated Sat, Mar 14, 2020 6:05 PM by kenneth