NLPBot Server Install

1. Install Python 3.5

During testing, Python 3.5 was installed as a standalone "alt-install" version on Linux.

Other approaches may work.

Login as ROOT

```
sudo su - root
```

Get source code for required version

```
wget https://www.python.org/ftp/python/3.5.3/Python-3.5.3.tgz
tar xzf Python-3.5.3.tgz
cd Python-3.5.3
```

Linux Dependencies

```
yum install openssl-devel
yum install zlib-devel bzip2-devel sqlite sqlite-devel openssl-devel
```

Configure and Compile as "ALTINSTALL"

```
./configure
make altinstall
```

Verify

```
$> python3.5 -V
Python 3.5.3
$> pip3.5 -V
pip 9.0.1 from /usr/local/lib/python3.5/site-packages (python 3.5)


ls -l /usr/local/bin
```

(shows location of alternative, newly installed Python)

2. Install pyMySQL Library

as `root`:

```
pip3.5 install pymysql
```

3. Install NLTK

as `root`:

```
pip3.5 install nltk
```

4. Install Machine Learning Libs

as `root`:

```
pip3.5 install numpy
pip3.5 install scipy
pip3.5 install pandas
pip3.5 install scikit-learn
```

5. Create the Linux BotUser

as `root`:

```
useradd botuser
```

6. Install GIT

as `root`:

```
yum install git
```

7. Install Java 8

as `root`: **Install Java 8**

- Download Java 8 as a zipped tar-ball from Oracle: https://java.com/en/download/help/linux_x64_install.xml#download
- Copy to the Server as root user
- Follow install instructions here: https://java.com/en/download/help/linux_x64_install.xml#install
- Rename Java root dir to appropriate location such as /usr/bin/java

8. MySQL Database Server Configuration

A MySQL database server is required to create the NLPBot database in. This can be a remote database or local to the botserver process.

The default database name for the NLPBot is

```
nlpbot
```

and the default user-name for the NLPBot to connect to the database server is

```
nlpbot
```

Database User Create a dedicated database for the NLPBot data-store.

As `root` user in MySQL, create a dedicated MySQL user for the NLPBot to connect to the datastore:

```
CREATE USER 'nlpbot'@'localhost' IDENTIFIED BY '<myPassword123>';

CREATE DATABASE nlpbot;

GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP, INDEX, ALTER on nlpbot.*
   TO 'nlpbot'@'localhost';

GRANT CREATE TEMPORARY TABLES  on nlpbot.*
   TO 'nlpbot'@'localhost';
```

The user-name must match the user configured in the NLPBot `config.ini` file (covered in the next section).

For authentication, replace `localhost` in the example above with the remote host-name that the ChatBot will be accessing the mySQL database *from*.

Set an appropriate password; this will need to be initialised (stored locally in encoded form) for the ChatBot later on (see the "Database Password Configuration" in the next section).

9. Install Bot Code and Configure

```
su - botuser

git clone https://github.com/edbullen/NLPBot.git
cd NLPBot
```

OR, to *pull updates* from GitHub to the local install (if the code just needs refreshing),

```
git pull
```

Ensure the following directories are in place:

```
~/NLPBot/config
~/NLPBot/dump
~/NLPBot/log
```

Database Password Configuration

Initialisation of Database Password for ChatBot

Password authentication details for the ChatBot to access the MySQL database are stored locally. To avoid storing the actual password in clear-text the password is stored in encoded form.

A separate file called

```
./config/.key
```

must be created in the config sub-directory with a single key phrase in it and no other text or characters - i.e.

```
$ cd config
$ echo "thisIsSecret" > ./.key
$ chmod 400 .key
```

(this is NOT the database password). Set appropriate file permissions on this file.

When the `.key` file is in place, initialise password access to the remote MySQL database using

```
cd ~/NLPBot
python3.5 pwdutil.py -s TheBotDatabasePassword
```

where "password" is the appropriate real password for the ChatBot MySQL database user account (so this is not the key phrase set above).

NOTE - the approach taken here is not very secure. It is better than storing the database password in clear text, but is still very limited and this configuration is not suitable for sensitive data.

Test Database Configuration A simple `pingDB.py` script is provided to test database access and base functionality.

The script connects as the configured mySQL user, creates a table "bot_test_tab", inserts a row, selects it back out and then drops the table.

Expected output is below:

```
$ python3.5 ./pingDB.py
Warning: (1051, "Unknown table 'nlpbot.bot_test_tab'")
  self._do_get_result()
execute drop_test_tab Args: None Response 0
execute create_test_tab Args: None Response 0
execute insert_test_tab Args: ('a', 1) Response 1
execute select_test_tab Args: 1 Response 1
execute drop_test_tab Args: None Response 0
```

Setup the Database Schema

Run the python script `setupDatabase.py` to create the database schema. EG:

```
$ python setupDatabase.py
Configuring Tables for database configuration:
        Server: localhost
        DB-User: nlpbot
        DB-Name: nlpbot
Continue? [Y/n] y
Connecting to database... connected.

Creating words table:
DROP TABLE IF EXISTS words
CREATE TABLE words ( hashid VARCHAR(16) COLLATE utf8_general_ci UNIQUE, word VARCHAR(64) COLLATE utf8_general_ci
UNIQUE )
...
(etc,etc)
...
Done.
```

10. Install the Stanford CoreNLP Package

as `botuser`:

```
su - botuser

cd ~
mkdir StanfordParser
cd  StanfordParser
```

```
wget http://nlp.stanford.edu/software/stanford-corenlp-full-2016-10-31.zip
unzip *.zip
```

Optional Path Simplification:

```
cd /home/botuser/StanfordParser
mv stanford-corenlp-full-2016-10-31 coreNLP
```

11. Configure the botuser ./config/config.ini file

Java

```
[Java]  #required for Stanford CoreNLP
bin: /usr/bin/java
```

Stanford NLP

```
[StanfordNLP]
corejar: /home/botuser/StanfordParser/coreNLP/stanford-corenlp-3.7.0.jar
modelsjar: /home/botuser/StanfordParser/coreNLP/stanford-corenlp-3.7.0-models.jar
```

NLTK Data DownLoad

As the **botuser** (not root):

```
$ python3.5
Python 3.5.3 (default, May  3 2017, 12:47:34)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-17)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
>>> nltk.download('punkt')
[nltk_data] Downloading package punkt to /home/botuser/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True

>>> nltk.download('averaged_perceptron_tagger')
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/botuser/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
True

>>> nltk.download("stopwords")
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/botuser/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Validate Database Connectivity

```
python3.5 pingDB.py
```

Re-Build the SciKit-Learn ML Model

Due to binary compatibility issues, usually the ML model must be re-built:

```
python3.5 mlClassGenerateRfModel.py
```

By default this reads in training data from `./analysis/featuresDump.csv` and writes it out to `./RFmodel.ml`
**Sample Output **

```
reading input from  ./analysis/featuresDump.csv
100 rows loaded
FEATURES = Index(['wordCount', 'stemmedCount', 'stemmedEndNN', 'CD', 'NN', 'NNP', 'NNPS',
       'NNS', 'PRP', 'VBG', 'VBZ', 'startTuple0', 'endTuple0', 'endTuple1',
       'endTuple2', 'verbBeforeNoun', 'qMark', 'qVerbCombo', 'qTripleScore',
       'sTripleScore'],
      dtype='object')

saving model to ./RFmodel.ml

Complete.
```

12. Start BotServer

```
nohup python3.5 botserver.py &
```

Logging

Bot server output is logged to

```
~/NLPBot/log/botserver.log
```

Stopping the Server

```
$ ps -ef | grep botserver
botuser  **22915** 22854  0 08:15 pts/0    00:00:01 python3.5 botserver.py
botuser  23165 22854  0 08:54 pts/0    00:00:00 grep botserver
```

```
kill -9 22915
```

Local Client Connect

```
python3.5 simpleclient.py -a localhost -p 9999
```

Remote Client Connect

Make sure the botserver port is allowed through the firewall.

```
python3.5 simpleclient.py -a 192.168.10.101 -p 9999
```

Debugging

SEt Logging

Set the **server** debug paramter to True in the `config.ini` file to increase the amount of logging information recorded by the botserver.py process.

```
[DEBUG]
assoc: False
weight: False
itemid: False
match: False
server: True
answer: False
```

Kill and restart the botserver process for the change to take effect.

Run in Single User Mode for Debugging

Extra diagnostics information and output from exceptions will be raised to (terminal) when run locally as follows:

```
python3.5 chatbot.py
```