README.md

MP3vec : A Reusable Machine-Constructed Feature Representation for Protein Sequences

Multi-Purpose Protein Prediction Vectors are generated by training a Deep Neural Network on a source problem (secondary structure prediction) and extracting the outputs of the intermediate layers. These vectors are transferable representations that can be used as baseline features across a wide range of target tasks involving sequence to sequence learning. This repository contains software tools to generate MP3 vectors for a dataset of given proteins. For more information, please see the project site at http://universe.bits-pilani.ac.in/goa/aduri/MP3vec.

Index

1 Package overview

The MP3vec python module has a core MP3Model class that wraps the trained model with a couple of utility functions to read and process input PSSM files. This model returns a (L,640) dimensional vector where L is the length of the protein sequence. For those users who wish to generate MP3vecs for PSSM files without using the python module, there is a command line utility script, **mp3vec**, to generate vectors for a list of PSSM files. Since the MP3vec generation requires PSSM files, another command line utility, **mp3pssm**, has been provided to generate them from a single FASTA file of sequences using PSI-BLAST.

This package has been tested on both Ubuntu 16.04 and Windows 10 and is fully compatible with Python 2.7x and 3.6x as well. Since the backend model has been created using TensorFlow, it can be accelerated on a GPU. However, the current install script only installs the CPU version of Tensorflow. Please note that if you wish to generate your own PSSMs using PSI-BLAST, you will need to install the NCBI BLAST+ software suite and download the Uniref90 dataset.

2 Installation of NCBI BLAST+ and Uniref90

2.1 Installation on Linux

2.1.1 Installing NCBI BLAST+

Download the Linux binaries for NCBI BLAST+ from **ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST** by copying the link in your browser's address bar. The Linux binaries will have a name like **ncbi-blast-2.7.1+-x64-linux.tar.gz** although the version number **2.7.1** may change over time.

Once the download is complete, copy the file to a folder of your choice, say `~/Desktop` and extract the file with the command `tar xvf ncbi-blast-2.7.1+-x64-linux.tar.gz`. All the BLAST+ executables will be stored in the folder `~/Desktop/ncbi-blast-2.7.1+/bin/`. However, you will not be able to run these programs from the command line unless you update the `PATH` environment variable. To do so, use the following command : `export PATH="~/Desktop/ncbi-blast-2.7.1+/bin:$PATH"`. If you run `echo $PATH`, you should now be able to see that the `PATH` variable has been updated. To make this change permanent, it is recommended that you update the `PATH` variable in your `~/.bashrc` file, otherwise the MP3vec utility scripts which generate PSSMs using PSI-BLAST will not be able to run `psiblast`. To do so, use your editor of choice to add the following line to your `~/.bashrc` file : `export PATH="~/Desktop/ncbi-blast-2.7.1+/bin:$PATH"`. Then reload `~/.bashrc` with the command `source ~/.bashrc`.

```
anket@GPU:~$ cp Downloads/ncbi-blast-2.7.1+-x64-linux.tar.gz Desktop/
sanket@GPU:~$ cd Desktop/
sanket@GPU:~/Desktop$ tar xvf ncbi-blast-2.7.1+-x64-linux.tar.gz
ncbi-blast-2.7.1+/
ncbi-blast-2.7.1+/doc/
ncbi-blast-2.7.1+/doc/README.txt
ncbi-blast-2.7.1+/LICENSE
ncbi-blast-2.7.1+/ChangeLog
ncbi-blast-2.7.1+/README
ncbi-blast-2.7.1+/ncbi_package_info
ncbi-blast-2.7.1+/bin/
ncbi-blast-2.7.1+/bin/makembindex
ncbi-blast-2.7.1+/bin/makeblastdb
ncbi-blast-2.7.1+/bin/tblastx
ncbi-blast-2.7.1+/bin/blastdbcheck
ncbi-blast-2.7.1+/bin/dustmasker
ncbi-blast-2.7.1+/bin/rpsblast
ncbi-blast-2.7.1+/bin/segmasker
ncbi-blast-2.7.1+/bin/windowmasker
ncbi-blast-2.7.1+/bin/convert2blastmask
ncbi-blast-2.7.1+/bin/update_blastdb.pl
ncbi-blast-2.7.1+/bin/psiblast
ncbi-blast-2.7.1+/bin/blastn
ncbi-blast-2.7.1+/bin/blast_formatter
ncbi-blast-2.7.1+/bin/blastp
ncbi-blast-2.7.1+/bin/makeprofiledb
ncbi-blast-2.7.1+/bin/legacy_blast.pl
ncbi-blast-2.7.1+/bin/blastdb_aliastool
ncbi-blast-2.7.1+/bin/deltablast
ncbi-blast-2.7.1+/bin/blastx
ncbi-blast-2.7.1+/bin/rpstblastn
ncbi-blast-2.7.1+/bin/tblastn
ncbi-blast-2.7.1+/bin/blastdbcmd
sanket@GPU:~/Desktop$ export PATH="/home/sanket/Desktop/ncbi-blast-2.7.1+/bin:$PATH"
sanket@GPU:~/Desktop$ echo $PATH
/home/sanket/Desktop/ncbi-blast-2.7.1+/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin
sanket@GPU:~/Desktop$ which psiblast
/home/sanket/Desktop/ncbi-blast-2.7.1+/bin/psiblast
sanket@GPU:~/Desktop$ psiblast -version
psiblast: 2.7.1+
 Package: blast 2.7.1, build Oct 18 2017 19:57:24
sanket@GPU:~/Desktop$
```

2.1.2 Setting up the Uniref90 database

You can find the download link for Uniref90 at https://www.uniprot.org/downloads. Make sure you download the FASTA file from **ftp://ftp.uniprot.org/pub/databases/uniprot/uniref/uniref90/uniref90.fasta.gz**. Create a new folder, say `~/Desktop/uniref90`, at a location of your choice and copy the `uniref90.fasta.gz` file to this folder. Extract the file in the folder using the command `gunzip ~/Desktop/uniref90/uniref90.fasta.gz` so that it appears at `~/Desktop/uniref90/uniref90.fasta`.

Now you can build the BLAST database using the `uniref90.fasta` file that you have extracted. The `makeblastdb` command is used to create the BLAST database. Open a Terminal and navigate to the folder containing the `uniref90.fasta` file, `~/Desktop/uniref90/`, and issue the command `makeblastdb -in uniref90.fasta -out uniref90.db -dbtype prot`. This creates a new database `uniref90.db` using the input `uniref90.fasta` file. The database type is specified by `-dbtype prot` which creates a protein database.

```
sanket@GPU:~$ mkdir ~/Desktop/uniref90
sanket@GPU:~$ cp ~/Downloads/uniref90.fasta.gz ~/Desktop/uniref90
sanket@GPU:~$ cd Desktop/uniref90/
sanket@GPU:~/Desktop/uniref90$ gunzip uniref90.fasta.gz
sanket@GPU:~/Desktop/uniref90$ makeblastdb -in uniref90.fasta -out uniref90.db -dbtype prot


Building a new DB, current time: 08/17/2018 00:23:46
New DB name:   /home/sanket/Desktop/uniref90/uniref90.db
New DB title:  uniref90.fasta
Sequence type: Protein
Keep MBits: T
Maximum file size: 1000000000B
Adding sequences from FASTA; added 52629880 sequences in 1536.75 seconds.
sanket@GPU:~/Desktop/uniref90$
```

The database has been successfully built and you can now run queries against it using `psiblast`. For ease of use, you need to set a new environment variable, `BLASTDB`, which points to the location of this database. You can do this with `export BLASTDB="/home/sanket/Desktop/uniref90/uniref90.db"`. Make sure that you do not accidentally enter the path to the original `uniref90.fasta` file, you don't need it any more. Consider adding this variable to your `~/.bashrc` file to make the changes permanent, similar to how you updated the `PATH` variable.

To check that the `BLASTDB` environment variable has been set properly, open a Terminal and type `echo $BLASTDB`. You should see the location of the database, i.e. `/home/sanket/Desktop/uniref90/uniref90.db`.

```
sanket@GPU:~$ export BLASTDB="/home/sanket/Desktop/uniref90/uniref90.db"
sanket@GPU:~$ echo $BLASTDB
/home/sanket/Desktop/uniref90/uniref90.db
sanket@GPU:~$
```

2.2 Installation on Windows

2.2.1 Installing NCBI BLAST+

Download the Windows Installer for NCBI BLAST+ from **ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST** by copying the link in your browser's address bar. The Windows Installer will have a name like **ncbi-blast-2.7.1+-win64.exe** although the version number **2.7.1** may change over time.

Once the download is complete, launch the installer. After you accept the license agreement, choose an installation destination for the BLAST+ programs. The default location is `C:\Program Files\NCBI\blast-2.7.1+`. Make a note of the installation location, we will need this later on. Click the `Install` button to complete the installation. The BLAST+ software suite should now be installed. Now you need to set the `Path` environment variable so that these programs can be invoked from the command line.

To set up the `Path` environment variable, go to `Control Panel > System and Security > System`. Then on the left, you'll find `Advanced system settings > Environment Variables`. You will see a list of User variables including one called `Path`. Select this variable and click on the `Edit...` button below and new window will pop up. Click `New` and then `Browse...`. Navigate to the location where you have installed the BLAST+ programs and select the `bin` folder. The complete path should look like `C:\Program Files\NCBI\blast-2.7.1+\bin` if you have installed the programs in the default location. Finally, click `OK` to save the changes you have made. The `Path` variable should now have been updated with the location of the `bin` folder which contains all the relevant programs.

To ensure that the `Path` variable has been set properly, open up a Command Prompt and type in `psiblast -version`. You should see the following :

```
C:\Users\Sanket>psiblast -version
psiblast: 2.7.1+
 Package: blast 2.7.1, build Oct 18 2017 19:55:35

C:\Users\Sanket>
```

2.2.2 Setting up the Uniref90 database

You can find the download link for Uniref90 at https://www.uniprot.org/downloads. Make sure you download the FASTA file from **ftp://ftp.uniprot.org/pub/databases/uniprot/uniref/uniref90/uniref90.fasta.gz**. Create a new folder, say `E:\uniref90`, at a location of your choice and copy the `uniref90.fasta.gz` file to this folder. Extract the file in the folder so that it appears at `E:\uniref90\uniref90.fasta`.

Now you can build the BLAST database using the `uniref90.fasta` file that you have extracted. The `makeblastdb` command is used to create the BLAST database. Open a Command Prompt and navigate to the folder containing the `uniref90.fasta` file, `E:\uniref90`, and issue the command `makeblastdb -in uniref90.fasta -out uniref90.db -dbtype prot`. This creates a new database `uniref90.db` using the input `uniref90.fasta` file. The database type is specified by `-dbtype prot` which creates a protein database.

```
E:\uniref90>makeblastdb -in uniref90.fasta -out uniref90.db -dbtype prot


Building a new DB, current time: 08/17/2018 00:22:55
New DB name:   E:\uniref90\uniref90.db
New DB title:  uniref90.fasta
Sequence type: Protein
Keep MBits: T
Maximum file size: 1000000000B
Adding sequences from FASTA; added 52629880 sequences in 1480.17 seconds.

E:\uniref90>
```

The database has been successfully built and you can now run queries against it using `psiblast`. For ease of use, you need to set a new environment variable, `BLASTDB`, which points to the location of this database. Go to `Control Panel > System and Security > System > Advanced system settings > Environment Variables` and create a new User variable. Set the variable name as `BLASTDB` and the variable value as the location of the database you have built, which in this case is `E:\uniref90\uniref90.db`. Make sure that you do not accidentally enter the path to the original `uniref90.fasta` file, you don't need it any more.

To check that the `BLASTDB` environment variable has been set properly, open a Command Prompt and type `echo %BLASTDB%`. You should see the location of the database, i.e. `E:\uniref90\uniref90.db`.

```
C:\Users\Sanket>echo %BLASTDB%
E:uniref90\uniref90.db

C:\Users\Sanket>
```

3. Installing the MP3vec package

3.1 Installation on Linux

You can download the MP3vec package as a zip file from https://github.com/sanketx/MP3vec/archive/master.zip. Alternatively, you can clone the repository using `git` from https://github.com/sanketx/MP3vec.git. Create a new directory, say `~/mp3_project` and clone the MP3vec repositorty or copy the zip file into this directory. If you're using the zip file, you can extract the contents using the `unzip` command. You can also rename the extracted folder from `MP3vec-master` to `MP3vec`.

3.1.1 Using the zip file

```
sanket@GPU:~$ mkdir mp3_project
sanket@GPU:~$ cd mp3_project/
sanket@GPU:~/mp3_project$ cp ~/Downloads/MP3vec-master.zip .
sanket@GPU:~/mp3_project$ unzip MP3vec-master.zip
Archive:  MP3vec-master.zip
2a8e3e6a764ed0e43256803c18f1be27c50435f3
   creating: MP3vec-master/
  inflating: MP3vec-master/MANIFEST.in
  inflating: MP3vec-master/README.md
   creating: MP3vec-master/mp3vec/
 extracting: MP3vec-master/mp3vec/__init__.py
  inflating: MP3vec-master/mp3vec/make_pssm.py
  inflating: MP3vec-master/mp3vec/mp3_model.h5
  inflating: MP3vec-master/mp3vec/mp3vec.py
  inflating: MP3vec-master/mp3vec/vectorize_directory.py
  inflating: MP3vec-master/setup.py
```

```
sanket@GPU:~/mp3_project$ mv MP3vec-master MP3vec
sanket@GPU:~/mp3_project$ ls MP3vec/
MANIFEST.in  mp3vec  README.md  setup.py
sanket@GPU:~/mp3_project$
```

3.1.2 Cloning the repository

```
sanket@GPU:~$ mkdir mp3_project
sanket@GPU:~$ cd mp3_project/
sanket@GPU:~/mp3_project$ git clone https://github.com/sanketx/MP3vec.git
Cloning into 'MP3vec'...
remote: Counting objects: 14, done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 14 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (14/14), done.
Checking connectivity... done.
sanket@GPU:~/mp3_project$ ls MP3vec/
MANIFEST.in  mp3vec  README.md  setup.py
sanket@GPU:~/mp3_project$
```

It is recommended that you use a virtual environment for the installation of this package. This creates a separate environment making it easier to manage packages and their dependencies. You will need `pip` to install it. If you don't have `pip`, follow the instructions at https://pip.pypa.io/en/stable/installing/ to install it. To install the virtual environment package, type `pip install virtualenv`.

```
sanket@GPU:~/mp3_project$ pip install virtualenv
Collecting virtualenv
  Downloading https://files.pythonhosted.org/packages/b6/30/96a02b2287098b23b875bc8c2f58071c35d2efe84f747b64d523721dc2b5/virtuale
    100% |████████████████████████████████| 1.9MB 618kB/s
Installing collected packages: virtualenv
Successfully installed virtualenv
sanket@GPU:~/mp3_project$
```

You can create a new virtual environment, say `mp3env` for installing the MP3vec package and its dependencies with `virtualenv mp3env`. If you wish to use a specific python version for creating the environment, specify the path to the desired python executable with the `-p` flag. For example, to use Python 3.x that comes with your system, you would type `virtualenv -p /usr/bin/python3 mp3env`. The same process works if you want to use Python 2.7 instead. Once the environment is created, it needs to be activated with `source mp3env/bin/activate`.

```
sanket@GPU:~/mp3_project$ virtualenv -p /usr/bin/python2.7 mp3env
Running virtualenv with interpreter /usr/bin/python2.7
New python executable in /home/sanket/mp3_project/mp3env/bin/python2.7
Also creating executable in /home/sanket/mp3_project/mp3env/bin/python
Installing setuptools, pip, wheel...done.
sanket@GPU:~/mp3_project$ source mp3env/bin/activate
(mp3env) sanket@GPU:~/mp3_project$
```

Now you can install the MP3vec package along with its dependencies in the virtual environment that we have just created with `pip install MP3vec/.`.

```
(mp3env) sanket@GPU:~/mp3_project$ pip install MP3vec/.
Processing /home/sanket/mp3_project/MP3vec
Collecting tensorflow>=1.8.0 (from mp3vec==0.0.1)
...(text ommited)
Successfully installed absl-py-0.4.0 astor-0.7.1 backports.weakref-1.0.post1 biopython-1.72 enum34-1.1.6 funcsigs-1.0.2 futures-3
(mp3env) sanket@GPU:~/mp3_project$
```

You can test the installation by opening a python shell and importing the module.

```
(mp3env) sanket@GPU:~/mp3_project$ python
Python 2.7.12 (default, Dec  4 2017, 14:50:18)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import mp3vec
Using TensorFlow backend.
```

```
>>> mp3vec.__version__
'0.0.1'
```

The package has now been successfully installed along with the **mp3vec** and **mp3pssm** utility scripts.

3.2 Installation on Windows

The first step is to install Python on Windows. You must install either version 3.5 or 3.6, since precompiled binaries for TensorFlow are currently not available for other versions of Python. You can get the Python Installer from https://www.python.org/downloads/windows/. Select the "Download Windows x86-64 executable installer" link for Python 3.6.6 and download the installer. Once it is downloaded, launch the installer. While installing, make sure you tick the "Add Python 3.6 to PATH" option before clicking on "Install Now". If you don't, you won't be able to launch Python from the Command Prompt. To test the install, open up the Command Prompt and type `python`

```
C:\Users\Sanket>python
Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 03:37:03) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

You can download the MP3vec package as a zip file from https://github.com/sanketx/MP3vec/archive/master.zip. Make a new folder, say `mp3_project` and copy the `MP3vec-master.zip` file to this new folder. Now extract the contents with `right-click > 7-zip > Extract here`. You should now have a new folder, `MP3vec-master`, inside `mp3_project` which contains 3 files, `MANIFEST.in`, `README.md`, and `setup.py` along with a folder `mp3vec`.

It is recommended that you use a virtual environment for the installation of this package. This creates a separate environment making it easier to manage packages and their dependencies. To install the virtual environment package, type `pip install virtualenv`.

You can create a new virtual environment, say `mp3env` for installing the MP3vec package and its dependencies with `virtualenv mp3env`. Once the environment is created, it needs to be activated with `mp3env\Scripts\activate`.

```
C:\Users\Sanket>cd mp3_project

C:\Users\Sanket\mp3_project>pip install virtualenv

C:\Users\Sanket\mp3_project>virtualenv mp3env
Using base prefix 'c:\\users\\sanket\\appdata\\local\\programs\\python\\python36'
New python executable in C:\Users\Sanket\mp3_project\mp3env\Scripts\python.exe
Installing setuptools, pip, wheel...done.

C:\Users\Sanket\mp3_project>mp3env\Scripts\activate

(mp3env) C:\Users\Sanket\mp3_project>
```

Now you can install the MP3vec package along with its dependencies in the virtual environment that we have just created with `pip install MP3vec-master\.`. Make sure that the `setup.py` and the `mp3vec` folder are present inside `MP3vec-master`, otherwise the installation will fail.

```
(mp3env) C:\Users\Sanket\mp3_project>pip install MP3vec-master\.
Processing c:\users\sanket\mp3_project\mp3vec-master
Collecting tensorflow>=1.8.0 (from mp3vec==0.0.1)
...(text ommited)
Successfully installed absl-py-0.4.0 astor-0.7.1 biopython-1.72 gast-0.2.0 grpcio-1.14.1 h5py-2.8.0 keras-2.2.2 keras-application

(mp3env) C:\Users\Sanket\mp3_project>
```

The installation is almost complete, however, TensorFlow will not run properly if `msvcp140.dll` isn't installed. To install it, download it from https://www.microsoft.com/en-us/download/details.aspx?id=53587 and run the exe to install the file. You can test the installation by opening a python shell and importing the module.

```
(mp3env) C:\Users\Sanket\mp3_project>python
Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 03:37:03) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import mp3vec
Using TensorFlow backend.
```

```
>>> mp3vec.__version__
'0.0.1'
>>>
```

4. Usage instructions

4.1 Command line script for generating MP3 Vectors

A command line utility, **mp3vec** has been provided for users who wish to generate MP3vecs without writing code. This utility is automatically installed when you install the mp3vec python package. You can specify a directory containing PSSM files and the script will vectorize them and write them to a destination directory. Currently, two output formats are supported, the binary numpy format with a .npy extension and the CSV format with a .csv extension. You need to specify the format while using the script. An optional model parameter is available in case you want to use a custom model, by default the script uses the pre-trained model provided in the package. You can view the flags with `mp3vec -h`.

```
(mp3env) sanket@GPU:~/mp3_project$ mp3vec -h
Using TensorFlow backend.
usage: mp3vec [-h] -i IN_DIRECTORY -o OUT_DIRECTORY [-m MODEL_FILE] -t
              {NPY,CSV}

MP3vec command line utility. This program can be used to generate Multi-
Purpose Protein Prediction vectors as described in the paper. It accepts PSSM
files as input and creates a protein feature vector that can be saved as
either a Numpy array or as a CSV file. To use this program, please provide the
path to a directory containing PSSM files. Ensure that these files have a
'.pssm' extension or they will be ignored. You also have to specify an output
directory where the generated vectors will be stored. Finally, you need to
specify the output file format, either numpy array or CSV file.

optional arguments:
  -h, --help            show this help message and exit
  -i IN_DIRECTORY, --in_directory IN_DIRECTORY
                        Path to Input Directory containing PSSM files
  -o OUT_DIRECTORY, --out_directory OUT_DIRECTORY
                        Path to Output Directory to write MP3vec files
  -m MODEL_FILE, --model_file MODEL_FILE
                        Path to MP3 model file. Optional parameter for custom
                        models
  -t {NPY,CSV}, --otype {NPY,CSV}
                        Output file format. numpy (NPY) or comma separated
                        values (CSV)
(mp3env) sanket@GPU:~/mp3_project$
```

4.2 Command line script for generating PSSM files using PSI-BLAST

The **mp3pssm** utility has been provided to ease the process of PSSM generation. It enables users to provide a single input fasta file with protein sequences and **mp3pssm** will automatically generate PSSMs for each sequence. You need to specify the input file, the destination directory for generated PSSM files, the BLAST database location (which you should have stored in the BLASTDB environment variable, if not, check the section "Installation of NCBI BLAST+ and Uniref90" for instructions. You can optionally specify the number of threads you wish to use in the BLAST search. You can view the flags with `mp3pssm -h`.

```
(mp3env) sanket@GPU:~/mp3_project$ mp3pssm -h
Using TensorFlow backend.
usage: mp3pssm [-h] -i IN_FILE -o OUT_DIRECTORY -d BLAST_DB [-n NUM_THREADS]

Wrapper utility for generating PSSM files using PSI-BLAST. This program
generates PSSM profiles for FASTA sequences. You need to provide the path to
an input FASTA file (multiple sequences are allowed) along with the
destination directory where the PSSM files will be stored. The files will be
named using the sequence ID in the FASTA file and will be saved with a '.pssm'
extension. You also need to specify the path to the Uniref90 BLAST database
and the number of threads you wish to use for the PSSM computation. By
default, only 1 thread will be used.

optional arguments:
```

```
  -h, --help            show this help message and exit
  -i IN_FILE, --in_file IN_FILE
                        Path to Input FASTA file containing protein sequences
  -o OUT_DIRECTORY, --out_directory OUT_DIRECTORY
                        Path to Output Directory to write PSSM files
  -d BLAST_DB, --blast_db BLAST_DB
                        Path to Uniref90 BLAST database
  -n NUM_THREADS, --num_threads NUM_THREADS
                        NNumber of threads (CPUs) to use in the BLAST search
(mp3env) sanket@GPU:~/mp3_project$
```

4.3 Example 1: Using the command line scripts to generate MP3 vectors

The following example takes you through the steps of using the command line scripts to generate MP3 vectors for a FASTA file with two protein sequences. The steps are nearly identical for both Linux and Windows platforms, and any differences will be highlighted.

First, copy the contents of the text box below and save them as a FASTA file, say `test.fa`, in the `mp3_project` folder. For Windows users, this folder will be `C:\Users\Sanket\mp3_project>`, for Linux users this will be the `~/mp3_project` directory.

```
>PROT1
MVKLTAELIEQAAQYTNAVRDRELDLRGYKIPVIENLGATLDQFDAIDFSDNEIRKLDGFPLLRRLKTLLVNNNRICRIG
EGLDQALPDLTELILTNNSLVELGDLDPLASLKSLTYLCILRNPVTNKKHYRLYVIYKVPQVRVLDFQKVKLKERQEAEK
MFKGKRGAQLAKDIAR
>PROT2
MDIRPNHTIYINNMNDKIKKEELKRSLYALFSQFGHVVDIVALKTMKMRGQAFVIFKELGSSTNALRQLQGFPFYGKPMR
IQYAKTDSDIISKMRG
```

You should now have a file `test.fa` in the `mp3_project` folder. Now make two new folders in the `mp3_project` folder, one called `pssm_dir` for storing the generated PSSM files, and another called `vec_dir` to store the generated MP3 vectors. Do this using the command `mkdir pssm_dir vec_dir`. Now you are ready to run the `mp3pssm` utility to generate the PSSM profiles for the protein sequences in `test.fa`.

Linux users should run the command `mp3pssm -i test.fa -o pssm_dir/ -d $BLASTDB -n 8`, while Windows users should run the command `mp3pssm -i test.fa -o pssm_dir/ -d %BLASTDB% -n 8`. The only difference is in the way the `BLASTDB` environment variable is provided. Make sure you have activated the virtual environment prior to running the script, otherwise you will get a `command not found / unrecognized command` error message. You can tell if the virtual environment is active by looking for the name of the environment enclosed in parentheses before the prompt, like `(mp3env) sanket@GPU:~$` or `(mp3env) C:\Users\Sanket>`.

```
(mp3env) sanket@GPU:~/mp3_project$ mkdir pssm_dir vec_dir
(mp3env) sanket@GPU:~/mp3_project$ ls
mp3env  MP3vec  pssm_dir  test.fa  vec_dir
(mp3env) sanket@GPU:~/mp3_project$ mp3pssm -i test.fa -o pssm_dir/ -d $BLASTDB -n 8
Using TensorFlow backend.
Generated PSSM for protein PROT1
Generated PSSM for protein PROT2
Generated PSSMs for 2 proteins
(mp3env) sanket@GPU:~/mp3_project$
```

You can check the contents of `pssm_dir` to see if the PSSM files have been generated. You should find `PROT1.pssm` and `PROT2.pssm` in the folder. Keep in mind that the file name of the generated file will be the same as the text provided in the FASTA file after the initial `>` symbol. The `.pssm` extension is automatically added to this name. In rare cases, running the PSI-BLAST query will not result in any hits, and no output PSSM file will be created. Unfortunately, MP3 vectors cannot be generated without the PSSM file.

Once the PSSM files have been generated and stored in `pssm_dir`, you can run the `mp3vec` command to generate MP3 vectors from these files. From within the `mp3_project` directory, run the command `mp3vec -i pssm_dir/ -o vec_dir/ -t NPY` to save the vectors as numpy files, or run `mp3vec -i pssm_dir/ -o vec_dir/ -t CSV` to save the vectors as CSV files. The output file names will the same as the input file names, only the extension will be different, either .npy or .csv depending on your choice of the output file type. Please note that the `mp3vec` script will only search the input folder for files ending with .pssm, any other files will be ignored.

```
(mp3env) sanket@GPU:~/mp3_project$ mp3vec -i pssm_dir/ -o vec_dir/ -t NPY
Using TensorFlow backend.
Vectorized PROT1, file 1 / 2
Vectorized PROT2, file 2 / 2
(mp3env) sanket@GPU:~/mp3_project$
(mp3env) sanket@GPU:~/mp3_project$ mp3vec -i pssm_dir/ -o vec_dir/ -t CSV
```

```
Using TensorFlow backend.
Vectorized PROT1, file 1 / 2
Vectorized PROT2, file 2 / 2
(mp3env) sanket@GPU:~/mp3_project$ ls vec_dir/
PROT1.csv  PROT1.npy  PROT2.csv  PROT2.npy
(mp3env) sanket@GPU:~/mp3_project$
```

Check the contents of `vec_dir`, you should be able to see the generated vector files. You can now use these vectors for your Machine Learning experiments. If you're using Python, you can load the numpy matrices directly. R and Matlab users can load the vectors from the csv files. Once you are done, you can deactivate the virtual environment by typing `deactivate` in the Terminal or Command Prompt.

4.4 Example 2: A Python script for generating MP3 vectors

If you would like to generate MP3 vectors using a python script, you can import the `mp3vec` package into your code. The `mp3vec` module has a core `MP3Model` class. The pretrained model provided with this package is used by default but you can specify a custom model by specifying the model file in the class constructor. The `vectorize()` function can be called on a numpy array containing the PSSM and the one-hot encoded protein sequence. The output of this function is the MP3 vector for that protein.

A utility function, `encode_file()`, is provided in order to read a PSSM file and convert it into a numpy array which can then be fed as input to the model. Note that this function automatically reads the protein sequence from the file and converts it to a one-hot encoded form. The function returns this sequence along with the protein matrix (one-hot vector + PSSM vec). The model's vectorize function can then be used to convert this protein matrix into the MP3 vector.

Make sure that the virtual environment where you have installed MP3vec is active. Once you are done, you can deactivate this environment.

```
from mp3vec import *
model = MP3Model()
seq, protein_matrix = encode_file("PROT1.pssm")
vec = model.vectorize(protein_matrix)
```