CA114 Project

What all ML needs

Train, Test and Validation Data

1. Introduction

Nowadays machine learning (ML) is a part of almost every big organisation. ML techniques are used to generate models that can make predictions or decisions. Such models are trained on large datasets (the exact size of such datasets depends on the task in question). Some examples use-cases of ML are: credit decisions, fraud prevention, process automation, personalised product placement and recommendations, machine translation, etc.

The training process of an ML algorithm typically interleaves three steps: a training, a validation and a testing step. During the training step, the model is adapted according to the data and its progress is validated (i.e. the validation step). The final model is then tested to check its generalisation abilities (i.e. the testing step). The data used in this whole process usually comes from three data sets: a training (or a train) set, a development or a validation set and a testing (or a test) set. The train, validation and test sets should be disjoint, i.e. they **should not overlap**.

Machine translation (MT) software is a fully automated software that can translate text in one language (the source) into text in another language (the target). Various approaches to tackle this task exist, the most common and recent ones are data-driven MT systems. Such systems rely on large amounts of parallel sentences. Parallel sentences are sentences in the source language, e.g. English, and their corresponding translation in the target language, e.g. German. This data allows for the ML algorithms behind MT to learn the most probable word or phrase translations so that when only a source sentence is given as input, the MT system can break it down into words or phrases and generate the most-probable translations. These words/phrases are then combined to form a complete translated sentence. Here is an example of parallel data:

Source - English (En)	**Target - German (De)**
Resumption of the session	Wiederaufnahme der Sitzungsperiode
You have requested a debate on this subject in the course of the next few days, during this part-session.	Im Parlament besteht der Wunsch nach einer Aussprache im Verlauf dieser Sitzungsperiode in den nächsten Tagen.
Please rise, then, for this minute' s silence.	Ich bitte Sie, sich zu einer Schweigeminute zu erheben.
(The House rose and observed a minute' s silence)	(Das Parlament erhebt sich zu einer Schweigeminute.)
Madam President, on a point of order.	Frau Präsidentin, zur Geschäftsordnung.
You will be aware from the press and television that there have been a number of bomb explosions and killings in Sri Lanka.	Wie Sie sicher aus der Presse und dem Fernsehen wissen, gab es in Sri Lanka mehrere Bombenexplosionen mit zahlreichen Toten.
...	...

The technique that has currently proven to be the most effective MT technique for a vast majority of translation tasks is based on artificial neural networks[1] and is referred to as neural machine translation (NMT)[2].

When working on MT, these datasets consist of parallel sentences (see example above). It is important to highlight that there is no overlap between the train, test and validation set. Typically in MT, one starts with a dataset of parallel sentences that is then split into train, validation and test sets. For example, if you would have 1 000 000 parallel sentences a typical split would retain 1 000 sentences for a validation set and 1 000 for a test set (thus using the remaining 998 000 parallel sentences as the train set).

[1] ANNs: https://towardsdatascience.com/introduction-to-artificial-neural-networks-ann-1aea15775ef9
[2] https://machinelearningmastery.com/introduction-neural-machine-translation/

2. Task

As you can see, data is very important for any ML task, including MT. As such, your task is to develop a *tool* that downloads parallel data from the internet and splits it into a train, validation and a test sets. As this tool is going to be part of a large MT framework it needs to be flexible, that is to provide various options that control its behaviour, to be self-contained, e.g. not to depend on hardcoded folders, links, etc., and responsive, that is, to notify the user of its status (e.g., if there is an error to provide a meaningful error message).

2.1. Specific requirements

This tool we should call the ***TTV component***. It should:

- be developed as one or more **shell scripts**. Only one should be the main/callable script.
- work in Linux
- receive *input* from the terminal
- *output* data in 6 files:
    - Source and target files for the train set
    - Source and target files for the test set
    - Source and target files for the validation set
- download data from the internet (this will be the data to be split). Within the scope of this task the link to use is:
  https://www.computing.dcu.ie/~dshterionov/Lectures/ca114/2019_2020_project/data
  which will contain data sets of parallel sentences in a single file in a tab-separated format, for example:
  https://www.computing.dcu.ie/~dshterionov/Lectures/ca114/2019_2020_project/data/project_example_EN_DE.csv
  The links to the files have the form:
  https://www.computing.dcu.ie/~dshterionov/Lectures/ca114/2019_2020_project/data/[SRC]-[TRG].csv
  Where [SRC] is a language code for the source side and the [TRG] is the language code for the target. The codes to be supported are **EN**, **DE**, **ES**, **FR**, **GA**, corresponding to English, German, Spanish, French and Irish (accordingly).
- not all combinations are available. There is parallel data only from and to English (EN). Your script should handle cases where the user requests data that does not exist, i.e. handle errors correctly.
- split the downloaded data into train, test and validation sets based on user-defined split sizes. That is, the user should specify the size of the test and the validation sets as a number of parallel sentences. The remaining would be left for the train set.
- allow for different ways of selecting the data. In particular, at least, but not limited to:
    - Random selection - sentences for the test and validation sets should be randomly selected from the downloaded data.
      Hint!: you can first rearrange the original (downloaded) data randomly and then take the first or last sentence pairs for your test and validation sets.
    - Length selection - the shortest or longest sentences should be in your test and/or validation set.
    - Based on keywords - read a set of keywords from a file and select parallel sentences that contain these keywords in the source sentences.
- output on the screen some statistics:
    - In case of success:
        - Which are the source and target languages
        - Which selection option was used
        - How many words are in each of the produced files
        - How many lines have been selected

        - In case of error:
            - A comprehensive error message.

2.2. Clarifications

- You can decide how the TTV component will accept options. For example, one solution is as input arguments (see below); another solution is as global variables.
- There are no restrictions on how to read or write files.
- There are no restrictions on which commands to use in your script. ***Hint!***: Use the ones we learned in class.
- All commands, scripts, operations and techniques that you will need to accomplish this task will be covered in class and the lab sessions. For your own and my convenience, I would highly recommend you to use the commands we have covered in the classes. This will confirm that you have grasped the taught material.

2.3. Some example invocations:

The following examples are simply for illustrative purposes. Your calls (invocations) can look differently and that is all right.

```
$ sh run_ttv.sh --help
```
Displays a help message to guide the user how to use your script.

```
$ sh run_ttv.sh --src-lang=EN --trg-lang=FR
```
Downloads English-French data and splits into train, test and validation sets based on your default options.

```
$ sh run_ttv.sh -s EN -t FR
```
The same as above, just uses short notation. That is, downloads English-French data and splits into train, test and validation sets based on your default options.

```
$ sh run_ttv.sh -s EN -t FR -l 1000 -k 2000
```
Downloads English-French data and splits into train, test and validation sets where your test set will contain 1000 parallel sentences and your validation set will contain 2000 parallel sentences.

```
$ sh run_ttv.sh -s EN -t FR --test-keywords='test_keywords.txt'
--val-keywords='val_keywords.txt'
```
Downloads English-French data and splits it into train, test and validation sets where your test set will contain parallel sentences that have the keywords from the `'test_keywords.txt'` file and your validation set will contain parallel sentences that have the keywords from `'val_keywords.txt'` file.

3. Guidelines

- You need to form a group of 3 people. If that is not possible, then of 4 people. Use the group system on the loop.
- Divide and conquer:
    - First, make a plan. Try to answer the following questions:
        - What is the general objective of the script?
        - What is the input and what is the output?
        - What are the different steps (do not write any code until these have been decided)?

        - What commands can do each of these steps or maybe you need to develop some extra scripts? Then what scripts need to be developed to do those steps?
        - What is the expected output of each step?
        - What can go wrong at each step?
        - What would be the interface?
        - Should you use pipes or files or both?
    - Decide who is going to do what? Maybe everyone can write some code individually and then combine it or maybe one can share ideas, others to code them and one to test the solution(s). Another way is to write code all together - one person is writing others are following their coding and verifying their solutions.
    - Write your code and test it multiple times.
        - You can create tests for each step (these are typically called unit tests) and tests for the complete script.
        - Have other people test your code
- Documentation:
    - Document what you do and put it in a report. This will be evaluated later on.
    - Try to focus on why you do what you do.
- Presentation:
    - At the end you would need to present your solution. It is not required to make a sophisticated powerpoint presentation. But you will need to showcase your solution so think about that too during your work.
- Do come to consult me. It is very crucial that you understand what is required and that you have a clear idea of what you need to do.

4. Evaluation

Here are the criteria for evaluating your work. Overall mark is 40% of your module grade[3]:

- Teamwork (10%):
    - How well did you collaborate with your colleagues?
    - What was your individual contribution?
- Code (15%):
    - Does your code work?
    - Is it well written?
    - Is it well organised?
    - Are there code comments?
    - Are there proper messages for success and error conditions?
- Report (5%):
    - You need to write a brief report that should reflect how you approached the project.
    - It should be no longer than 2 pages A4 format.
    - It should simply reflect your progress and decisions from the beginning to the end.
- Presentation (10%):
    - Can you convince me that your TTV component does what it is supposed to?
    - Can you convince me that you did a great job?

5. Deadline: **05 Apr. 2020, 23:59**

[3] The other 10% of your continuous assessment comes from your lab attendance.