Sem. 1 FINAL

Python Env.

Requirements	Why?
Python3 SciPy env.	Fundamental algorithms for scientific computing in Python, offering algorithms covering optimization, integration, interpolation, eigenvalue problems, algebraic equations, differential equations, statistics and many other classes of problems.
Keras (with a TensorFlow or Theano backend)	Deep learning API written in Python (hence its compatibility), running on top of the machine learning platform TensorFlow. Developed with a focus on enabling fast experimentation. Keras is the high-level API of TensorFlow2 (essentially an infrastructure level for differentiable programming)
NumPy	Brings the computational power of languages like C and Fortran to Python. Offers unique powerful *n*-dimensional arrays and algorithms in numerical computing and vectorization.
Matplotlib	A comprehensive library for creating static, animated, and interactive visualizations in Python.

Imports

`string`

- This built in `string` class provides the ability to do complex variable substitutions and value formatting via the *format()* method. Furthermore, the editable *Formatter()* class allows you to create and customize your own string formatting behaviors.

> **Importance**
>
> Especially relevant in encoding and developing patterns between such strings from imported and learned data.

`re`

- Provides regular expression matching operations similar to those found in Perl.
- A regular expression (or RE) specifies a set of strings that matches it; the functions in this module let you check if a particular string matches a given regular expression (or if a given regular expression matches a particular string, which comes down to the same thing).
- Regular expressions can be concatenated to form new regular expressions; if *A* and *B* are both regular expressions, then *AB* is also a regular expression. In general, if a string *p* matches *A* and another string *q* matches *B*, the string *pq* will match AB. This holds unless *A* or *B* contain low

precedence operations; boundary conditions between *A* and *B*; or have numbered group references. Thus, complex expressions can easily be constructed from simpler primitive expressions like the ones described here.

> **Importance**
>
> As we are dealing with strings, which can be each represented distinctly by a regex expression, regular expressions matching operations are especially important to find relations and patterns between data.

`pickle`

- The `pickle` module implements binary protocols for serializing and de-serializing a Python object structure. "*Pickling*" is the process whereby a Python object hierarchy is converted into a byte stream, and "*unpickling*" is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy.
- The data format used by `pickle` is Python-specific. This has the advantage that there are no restrictions imposed by external standards such as JSON or XDR (which can't represent pointer sharing); however it means that non-Python programs may not be able to reconstruct pickled Python objects.
- By default, the `pickle` data format uses a relatively compact binary representation. If you need optimal size characteristics, you can efficiently compress pickled data.

> **Importance**
>
> Pickling and unpickling is essentially what it means to *sort*. However, with this operation being as specific as translating a language with a specific large data set, the **efficacy** AND the **efficiency** are especially important. *pickle* does exactly that for us, efficiently converting our objects of data into byte streams, allowing to save your ML models, to minimize lengthy re-training and allows sharing, committing, and re-loading pre-trained machine learning models.

`unicodedata`

- This module provides access to the Unicode Character Database (UCD) which defines character properties for all Unicode characters. The data contained in this database is compiled from the UCD version 14.0.0.
- Python's string type uses the Unicode Standard for representing characters, which lets Python programs work with all these different possible characters. Unicode is a specification that aims to list every character used by human languages and give each character its own unique code.

> **Importance**
>
> Similar to *re* and *string*, the functionality to represent certain properties of a string/text is important in evaulating patterns in ML.

`numpy`

- Functionality mentioned above

> **Importance**
>
> *NumPy* aims to provide an array object that is up to 50x faster than traditional Python lists, important for handling large data sets filled with their own arrays and sets of data.

`pandas`

- `pandas` is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python.
- Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named NumPy, which provides support for multidimensional arrays.

> **Importance**
>
> Data in *pandas* is often used to feed statistical analysis in *SciPy*, plotting functions from *Matplotlib*, and machine learning algorithms in *Scikit-learn*. Alongside these environments, *pandas* is essential for data analysis and data representation.

`google-colab (google.colab)`

- Colab notebooks allow you to combine executable code and rich text in a single document, along with images, *HTML*, *LaTeX* and more.
- With Colab, it's able to importean image dataset, train an image classifier on it, and evaluate the model, all in just a few lines of code. Colab notebooks execute code on Google's cloud servers, allowing the leverage the power of Google hardware, including GPUs and TPUs.
- Colab is used extensively in the machine learning community with applications including:
  - Getting started with TensorFlow
  - Developing and training neural networks
  - Experimenting with TPUs
  - Disseminating AI research
  - Creating tutorials

> **Importance**
>
> Although it wasn't necessary to use Google Colab itself, the module provided allowed to import data provided and downloaded, which contained translated Latin words and sentences.

`keras.{function}`

- Functionality mentioned above

> Importance
>
> Keras makes deep learning accessible and local, providing a minimal approach to run neural networks. As this project wasn't a full-scale deep-learning machine learning one, Keras was the most optimal to even attempt this project.

`nltk.translate.bleu_score`

BLEU is a precision based metric; to emulate recall, the brevity penalty (BP) is introduced to compensate for the possibility of high precision translation that are too short. The BP is calculated as:

$$BP = \begin{cases} 1 & \text{if} \quad c > r \\ e^{1-r/c} & \text{if} \quad c \leq r \end{cases} \qquad (2)$$

where c and r respectively refers to the length of the hypothesis translations and the reference translations. The resulting system BLEU score is calculated as follows:

$$\text{BLEU} = \text{BP} \times \exp(\sum_{n=1}^{N} w_n \log p_n) \qquad (3)$$

where n refers to the orders of n-gram considered for p_n and w_n refers to the weights assigned for the n-gram precisions; in practice, the weights are uniformly distributed.

A BLEU score can range from 0 to 1 and the closer to 1 indicates that a hypothesis translation is closer to the reference translation[2].

> Importance

Very simply stated, BLEU is a quality metric score for MT systems that attempts to measure the correspondence between a machine translation output and a human translation. The central idea behind BLEU is that **the closer a machine translation is to a professional human translation, the better it is**.

Dataset

- The dataset used for this project was provided by Metatext at see datasets.