Sem. 1 FINAL

Why is Translating Latin So Hard?

If there's one thing that everyone who's studied Latin could agree on, it's that the grammar rules are incredibly hard. The word order is arbitrary, each of the verbs has several cases and all the nouns have gender. Latin has a very complex grammar including cases, genders, declensions and vastly different verb forms for different tenses (of which there are many). For example, nouns are divided into (main) 3 declensions unimaginatively named 1st, 2nd, and 3rd. There are masculine, feminine and neuter genders and 6 cases determined by what role the noun plays in the sentence; nominative, accusative, genitive, dative ablative, locative and vocative (7 if you include vocative). The noun will have a different ending depending on each of these pieces of information. Although these endings are not unique, that gives 54 different combinations.

Verbs can be very irregular. Verbs belong to one of 3 conjugations (-are, -ere and -ire) and may be regular or irregular. The verb endings for I, you (singular), he/she/it, we, you (plural) and they, are all different, and change again for different tenses. So while "to walk" has just 4 endings (walk, walks, walked, and walking), Latin verbs can run into 30–40. As a result, it is not enough to simply learn the stem or the infinitive ("carry", "to carry"). You must learn the present tense first-person singular form, the infinitive, the first-person singular imperfect and past participle together ("fero", "ferre", "tuli", "latum").

Latin was incredibly influential on Romance languages (Spanish, French, Portuguese among others) and therefore Latin will allow you to effectively understand these languages when written as well as, to a lesser extent, spoken. Many people don't realize the prevalence of Latin in English either, law is full of such references ("habeas corpus", "mens rea") as well as idioms ("status quo", "ad infinitum")

Significant Latin Literature

- ***Saint Augustine's Confessions*** **by Saint Augustine**
  - Augustine was a gifted teacher who abandoned his secular career and eventually became bishop of Hippo. His Confessions are a remarkable record of his wrestlings to accept his faith, his struggles to overcome sexual desire and renounce marriage and ambition. His final moment of conversion in a Milan garden is deeply moving.
- ***On Obligations*** **by Cicero**
  - The great Roman statesman Cicero lived at the center of power. He was an advocate and orator as well as philosopher, who met his death bravely at the hands of Mark Antony's executioners. *On Obligations* was written after the assassination of Julius Caesar to provide principles of behavior for aspiring politicians. Exploring as it does the tensions between honorable conduct and expediency in public life, it should be recommended reading for all public servants.
- ***The Rise of Rome*** **by Livy**
  - The Roman historian Livy wrote a massive history of Rome in 142 books, of which only 35 survive in their entirety. In the first five books, translated here, he covers the period from Rome's beginnings to her first major defeat, by the Gauls, in 390 BC. Among the many stories he includes are Romulus and Remus, the rape of Lucretia, Horatius at the bridge, and Cincinnatus called from his farm to save the state.
- ***On the Nature of the Universe*** **by Lucretius**
  - Lucretius lived during the collapse of the Roman republic, and his poem *De rerum natura* sets out to relieve men of a fear of death. He argues that the world and everything in it are governed by the laws of nature, not by the gods, and the soul cannot be punished after death because it is mortal, and dies with the body. The book is an astonishing mix of scientific treatise, moral tract, and wonderful poetry.
- ***Meditations*** **by Marcus Aurelius**
  - Roman emperor Marcus Aurelius was probably on military campaign in Germany when he wrote his philosophical reflections in a private notebook. Drawing on Stoic teachings, particularly those of Epictetus, Marcus tried to summarize the principles by which he led his life, to help to make sense of death and to look for

moral significance in the natural world. Intimate writings, they bring us close to the personality of the emperor, who is often disillusioned with his own status, and with human life in general.

- ***Metamorphoses*** **by Ovid**
    - The *Metamorphoses* is a wonderful collection of legendary stories and myth, often involving transformation, beginning with the transformation of Chaos into an ordered universe. In witty and elegant verse Ovid narrates the stories of Echo and Narcissus, Pyramus and Thisbe, Perseus and Andromeda, the rape of Proserpine, Orpheus and Eurydice, and many more.
- ***Agricola*** **and** ***Germany*** **by Tacitus**
    - Tacitus is perhaps best known for the *Histories* and the *Annals,* an account of life under emperors Tiberius, Claudius and Nero. The shorter *Agricola* and *Germany* consist of a life of his father-in-law, who completed the conquest of Britain, and an account of Rome's most dangerous enemies, the Germans. They are fascinating accounts of the two countries and their people, the northern 'barbarians'. Later, German nationalists attempted to appropriate *Germania* in support of National Socialist racial ideas.
- ***Georgics*** **by Virgil**
    - The *Georgics* is a poem of celebration for the land and the farmer's life. Virgil doesn't romanticize, rather he describes the setbacks as well as the rewards of working the land, and provides memorable descriptions of vine and olive cultivation, raising crops, and bee-keeping. It is both a practical agricultural manual and allegory, and brings the ancient rural world vividly to life.
- ***Aeneid*** **by Virgil**
    - The story of Aeneas' seven-year journey from the ruins of Troy to Italy, where he becomes the founding ancestor of Rome, is a narrative on an epic scale. Not only do Aeneas and his companions have to contend with the natural elements, they are at the whim of the gods and goddesses who hamper and assist them. It tells of Aeneas' love affair with Dido of Carthage and of Aeneas' encounters with the Harpies and the Cumaean Sibyl, and his adventures in the Underworld.

Python Environment

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 evaluating 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 (**B**i**L**ingual **E**valuation **U**nderstudy) is a metric for automatically evaluating machine-translated text. The BLEU score is a number between zero and one that measures the similarity of the machine-translated text to a set of high quality reference translations. A value of 0 means that the machine-translated output has no overlap with the reference translation (low quality) while a value of 1 means there is perfect overlap with the reference translations (high quality).
- It has been shown that BLEU scores correlate well with human judgment of translation quality. Note that even human translators do not achieve a perfect score of 1.0.

> Mathematical Details

1. BLEU Formula with $i - gram$ count

$$\text{BLEU} = \underbrace{\min\Big(1, \exp\big(1 - \frac{\text{reference-length}}{\text{output-length}}\big)\Big)}_{\text{brevity penalty}} \underbrace{\Big(\prod_{i=1}^{4} precision_i\Big)^{1/4}}_{\text{n-gram overlap}}$$

$$precision_i = \frac{\sum_{\text{snt}\in\text{Cand-Corpus}} \sum_{i\in\text{snt}} \min(m^i_{cand}, m^i_{ref})}{w^i_t = \sum_{\text{snt'}\in\text{Cand-Corpus}} \sum_{i'\in\text{snt'}} m^{i'}_{cand}}$$

m^i_{cand} = is the count of $i - gram$ in candidate matching the reference translation
m^i_{ref} = the count of $i - gram$ in the reference translation
w^i_t = the total number of $i - grams$ in candidate translation

The formula consists of two parts: the brevity penalty and the reference translation:

- **Brevity Penality**
  - The brevity penalty penalizes generated translations that are too short compared to the closest reference length, with an exponential decay. The brevity penalty compensates for the fact that the BLEU score has no recall term.
- **N-Gram Overlap**
  - The n-gram overlap counts how many unigrams, bigrams, trigrams, and four-grams (*i*=1,...,4) match their n-gram counterpart in the reference translations. This term acts as a precision metric. Unigrams account for *adequacy*, while longer n-grams account for *fluency* of the translation. To avoid overcounting, the n-gram counts are clipped to the maximal n-gram count occurring in the reference (m^n_{ref}).

2. BP → BLEU

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

c = length of the hypothesis translations
r = reference translations

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

n = the orders of n-gram considered for p_n
w_n = the weights associated to the n-gram precisions; in practice, the weights are uniformly distributed

Importance

BLEU is a quality metric score for MT systems that attempts to measure the correspondence between a machine translation output and a human translation. It is essentially the essence of the project and used to compare English translations to Latin.

Latin Dataset

- The dataset used for this project was provided by Metatext at see datasets.
- Other datasets have been considered: Lexilogos, The Latin Library, PHI Latin Texts, and a public Github Latin dataset.