NLP Course Useful Material

Andrea Galassi, Federico Ruggeri

October 18, 2021

Abstract

This document presents a collection of useful material for the Natural Language Processing course (held by Prof. Torroni). We hope you find this document useful!

1 Introduction

We report useful material that might be of use throughout the course. The provided material is mainly centred on python programming libraries to be used during assignments/projects. Additionally, we also report other useful recommendations, such as course best practices and writing guidelines with Latex. Python libraries are grouped up based on their domain of application, such as data visualization, machine learning, deep learning and so on. We highly recommend you to have a look and have at least a basic understanding of all these libraries. These libraries offer a variety of useful APIs.

Do not reinvent the wheel and rely on existing material (if possible)!

In this way, your work becomes significantly less error-prone and you can focus on the core part of each assignment/project.

2 Python programming libraries

We report an extensive list of useful and well-known python programming libraries. All of them provide detailed documentation: do not worry if you have never tried one of them, there's lot of online material enough for you to quickly get a good grasp of the library. StackOverflow is one of your best friends. Also remember that you can always contact us!

2.1 Data Management

Data management covers a wide variety of steps, from data loading to data organization and usage.

1. **Pandas**: handling spreadsheets in a very efficient way.

2. **JSON**: simple library for handling JSON files. Check also simplejson python library (faster and more up to date).

3. **Pickle**: python data serialization. For weakref objects you might also need to install dill.

4. **NumPy**: It is particularly efficient and lightweight when handling numerical data and some python structures like dictionaries. It is your way to go when you need to store matrices or heavy numerical objects.

2.2 Text preprocessing

Text preprocessing is one of the core parts of a NLP pipeline. Remember that having good data covers around 90% of the work. Always make sure your cleaning and text normalization steps are doing what they are supposed to do!

1. **NLTK**: a huge suite of tools, spanning from tokenization to tree parsing.

2. **CoreNLP**: high quality text library from stanford. It is mainly centred on tokenization and parsing. The library follows the client/server architecture. Check the python client wrapper if you want to use it!

3. **Gensim**: a lot of functionalities and especially useful for loading non-contextual embedding models.

4. **TextBlob**: Part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

5. **SpaCy**: very fast and with many functionalities in common with NLTK and CoreNLP.

6. **Polyglot**: supports text preprocessing pipelines in many different languages.

7. **Regular Expressions**: there's always that one person who likes writing regex and forgets about them the day after.

2.3 Numerical Operations

It is very important to define efficient numerical operations to avoid computation bottlenecks. Make use of existing libraries (and their APIs) to define fast and elegant code.

1. **Numpy**: one of the most used numerical libraries, it is usually integrated with other well-known libraries (e.g. Tensorflow and Pytorch for what concerns deep learning).

2. **Pandas**: it offers useful numerical operations regarding stored data.

3. **Scipy**: includes modules for linear algebra, optimization, integration, special functions, signal and image processing, statistics, genetic algorithms, ODE solvers, and others.

2.4 Machine Learning

'Classic' machine learning methods and statistical operations always represent one of the initial steps of the experimental setup. It is not always necessary to directly employ heavy deep learning models for a certain problem. These models also represent strong baselines that might be worth exploring.

1. **Scikit-learn**: a suite of functionalities, mainly concerning machine learning models. It directly other well-known libraries like Numpy.

2.5 Deep Learning

The definition of deep learning models can be done at different levels. Usually, you don't want to define basic models like feedforward neural networks from scratch. Thus, there exist high level libraries that act as wrappers to more low level ones. The following list tries only reports the most common deep learning libraries for which we have some experience. The main advantage of using well-known and broadly used deep learning libraries is that they are guaranteed to support new models in the future without too much delay.

1. **Tensorflow**: TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.
2. **Pytorch**: An open source machine learning framework that accelerates the path from research prototyping to production deployment. Pytorch follows a slightly different formulation with respect to tensorflow. If you are used to tensorflow (and viceversa) you might need some time to get used to it. It has become, along with tensorflow, one of the most used deep learning libraries nowadays.
3. **Keras**: a high-level deep learning library that can be used on the top of other libraries (e.g. Tensorflow). It is one the best libraries to getting started to deep learning.

   We highly recommend you to start from here!

2.6 Data Visualization

One of the most underestimated steps is definitely data visualization. A good and clean visualization of some aspects of your problem can be of great help. Additionally, when presenting your work, e.g. in a written report or scientific article, it is very important to define the best concise and self-explanatory way to show your relevant results.

1. **Matplotlib**: the basic and well-known data visualization library for python. It supports a wide variety of visualization methods and, usually, it is what you need in the majority of cases.

2. **Seaborn**: a high level visualization library that is built on top of Matplotlib to achieve equal or better results in fewer lines of code.

3. **GGplot**: Python implementation of the R visualization library.

4. **Plotly**: A very useful library that also supports interactive plots.

5. **Bokeh**: Another interesting library that directly supports interactive plots. Bokeh supports unique visualizations like Geospatial plots, Network graphs, etc. right out of the box.

3 Practice Environment (Google Colab)

We'll use Google colab for tutorials and assignments throughout the course. You might had already some practice with colab in past courses. In any case, **this** is a valid resource for a quick recap or first introduction. Knowing how to work on google colab is a mandatory requirement of the course, thus, make sure you are quite comfortable with the tool. In case of issues, doubts or questions, feel free to contact us (see next section for contacting).

4 Course best practices and recommendations

This section probably represents the hardest challenge to the majority of students (yes, you!). To avoid possible issues during the course, we thought about writing down some useful tips. These recommendations are few and very simple to understand, thus, make sure you give a look at them at least once! Jokes aside, we try our best to reduce learning overhead and possible related difficulties to let you focus entirely on the content of the course.

1. **Contacting**: Always hit the 'Reply All' button when replying to one of our emails. Vice-versa, remember to include of all us when writing down an email. This allows us to keep updated on all of you and reduce the number of emails.

2. **Forum**: The course website (virtuale) also supports a dedicated forum section. In this section you can look for team members (assignments, projects), but you can also post any kind of question/doubt. We highly recommend using the forum (even for asking course related questions). In this way, we have a centralized system that reduces the risk of missing your message, as well as, avoids duplicate questions.

   Don't be shy! Everything that happens in the course forum remains in the course forum!

3. **Projects/Thesis**: If you really the course (or one of us), you can visit our **personal research group website**. In the website, you can find a complete and extensive list of research topics for project activities, thesis

and so on. In this way, you can start having a general idea of what we do. Subsequently, feel free to contact us via email to schedule a meeting!

5 Writing a good report in Latex

In this section, we briefly give you some suggestions about writing a good and concise report. We highly recommend you to write your report in Latex since it has become the standard for scientific publications and much more. This section is not a Latex tutorial, thus, we redirect you to well-known **tutorials**. Lastly, Latex can be used either offline or online depending on the tool you are using. Our personal recommendation is Overleaf. It is an online Latex platform that avoids the troublesome problem of package management. Alternatively, there are offline solutions, i.e. editors, like **Texstudio**. Feel free to use any of these solutions based on your preference.

Back to the main question, what kind of reports do we expect? Here are some useful tips that might simplify your life and give you a more concrete idea.

1. **Organization**: A precise and clean content organization represents a crucial part when writing down your work. Define a general introduction so that the reader gets an idea of what you are going to present in more detail. Give sufficient background and proceed on describing your results. Last but not least, motivate each of your actions and analyse your results.

2. **Writing**: Avoid making tutorials of tools you use, we know them! For example, it is not important that you describe a deep learning model in its entirety, step by step. Give a general background and focus on those architecture parts (if any) that are relevant to your work. Remember that your results are the core part of the work. Make sure to highlight your contribution!

3. **Visualization**: A good and self-explanatory data visualization strongly emphasizes your contribution. Avoid reporting code or in-editor screenshots! Prefer defining a good table or plot that sums up your main message!

4. **Read and take inspiration**: Read other people work (scientific publications and reports) to check the recommended way to report your findings! The more you read, the better you become.