Management of Multimedia Information in Internet

Module 5. Natural Language Processing (NLP)

Unit 1. Introduction to NLP

Boni García

http://bonigarcia.github.io/
boni.garcia@uc3m.es

Telematic Engineering Department
School of Engineering

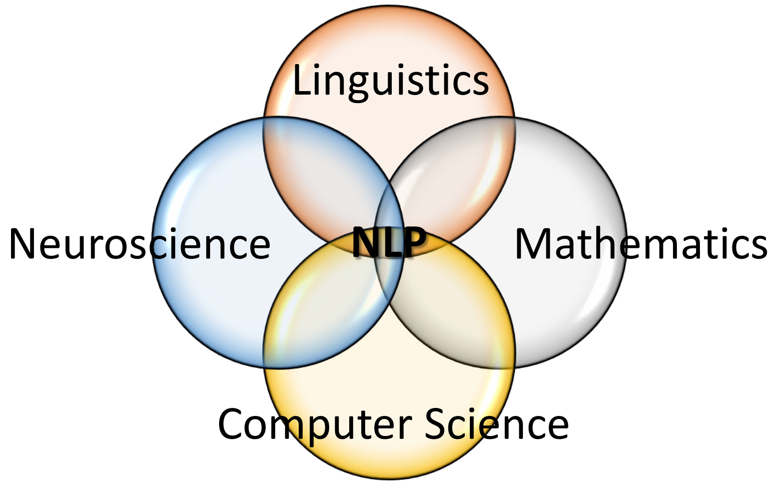
2020/2021

uc3m | Universidad Carlos III de Madrid

Table of contents

1. Introduction

- **Natural Language Processing** (**NLP**) is an interdisciplinary discipline concerned with the interactions between computers and human natural languages (such as English, Spanish, etc.)
  - NLP is sometimes categorized as a subfield of Artificial Intelligence (**AI**)
  - AI is a branch of computer science concerned with building smarts systems (such as expert systems or intelligent agents) capable of performing tasks that typically require human intelligence

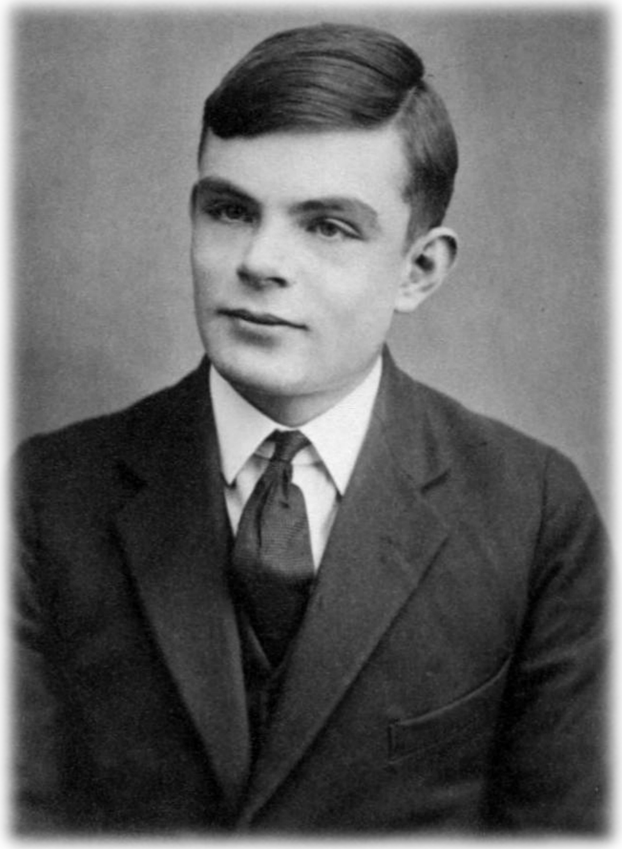


1. Introduction

- NLP has its roots in the 1950s. At that time, Alan Turing published an article titled "Computing Machinery and Intelligence" which proposed what is now called the **Turing test** as a criterion of intelligence

" A computer would deserve to be called intelligent if it could deceive a human into believing that it was human.

- *Alan Turing*



Source: Wikipedia
https://en.wikipedia.org/wiki/Alan_Turing

1. Introduction

- NLP applications must be able to **understand**, to some extent, a natural language
- This generally involves converting natural language (text) into data (numbers) that a computer can use
- The ultimate goal is to achieve human-like comprehension of natural languages

Table of contents

2. NLP applications

- Some of the most common NLP applications are:
  - **Text classification**
    - Assign predefined tags to a text
    - An example of text classification is the **spam detection**. In this case, text is classified as *spam* or *ham* (binary classifier)
    - Other example is **sentiment analysis**, i.e. identifying subjective information in texts, typically understanding if it is positive, neutral, or negative (multi-label classifier)
  - **Question Answering (QA)**
    - Systems that automatically answer questions posed by humans in a natural language
    - QA systems typically uses a huge knowledge base to deal with a range of questions
    - Example: IBM’s Watson
  - **Chatbots**
    - Application used to conduct an on-line chat conversation via text or text-to-speech
    - Chatbots typically use canned and heuristic responses based on that pattern matching

2. NLP applications

- **Conversational Agents (CA)** (also known as virtual assistants)
  - System capable of having a coherent conversation with a human
  - It can present a text-only interface or be a spoken dialog system
  - Examples: Apple Siri, Google Assistant, Microsoft Cortana, or Amazon Alexa
- **Machine Translation (MT)**
  - Translate a given text from a language to another
  - For example: Google Translate
- **Named Entity Recognition (NER)** (also known as text extraction)
  - Automatic detection of specific information in a text, such as names, companies, places, etc.
- **Auto correct**
  - Grammar checking software and auto-correct functions. Example: Grammarly
- **Automatic summarization**
  - Reduce a text retaining the most important information (abstract)

Table of contents

3. NLP approaches

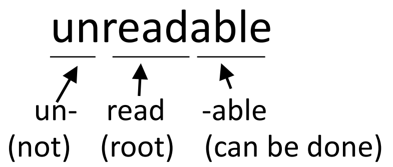
- NLP applications is carried out using different approaches:

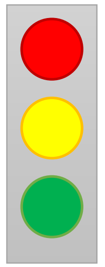
1. **Symbolic** NLP (1950s - early 1990s)
   - Based on human-developed grammar **rules** and lexicons to process text and model different language phenomena
2. **Statistical** NLP (1990s - 2010s)
   - Based of **Machine Learning** (ML) algorithms for language processing
   - Use of extensive sets of text documents (**corpus**) to automatically find frequent patterns of linguistic phenomena
3. **Neural** NLP (nowadays)
   - Based on **Artificial Neural Networks** (ANN) and **Deep Learning** (DL)

Table of contents

4. Symbolic NLP

- Symbolic NLP is based on human-developed grammar rules and lexicons (vocabulary) to process text
- It deals with different aspects of linguistics:
  - **Phonology**: study how languages systematically organize their sounds (or signs, in sign languages) $\longrightarrow$ [aɪ p^{h}iː eɪ]
  - **Morphology**: study of words, how they are formed, and their relationship to other words $\longrightarrow$
  - **Syntax**: set of rules, principles, and processes that govern the structure of sentences in a given language, usually including word order
  - **Semantics**: study of meaning
  - **Pragmatics**: study the ways in which context contributes to meaning

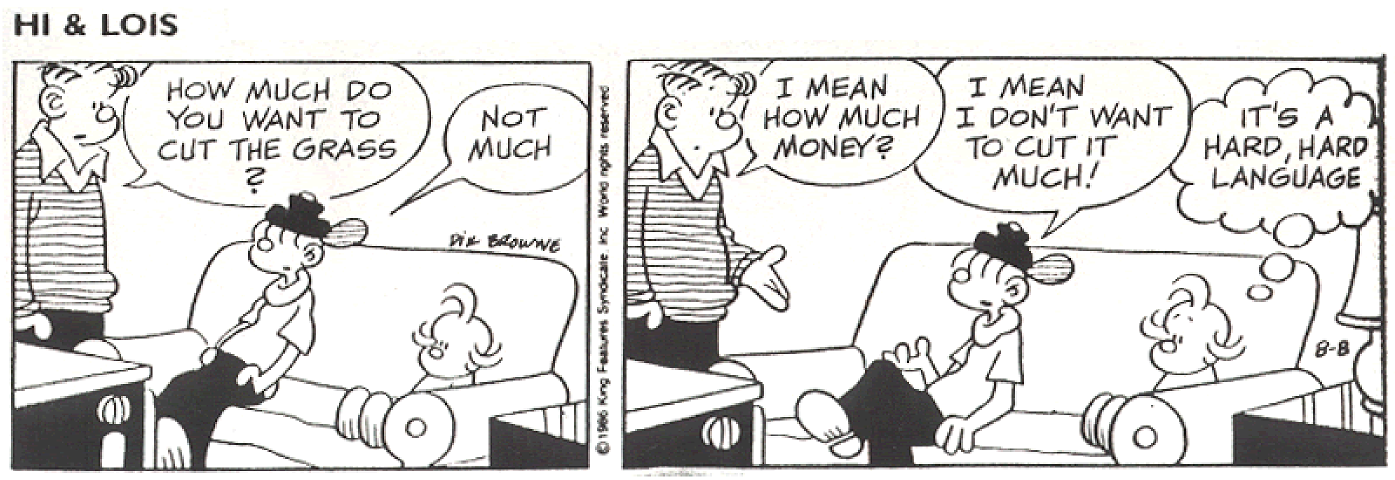




- **Syntax:** Green (bottom); yellow (middle); red (top)
- **Semantics**: Green=go; yellow= warning; red=stop
- **Pragmatics**: If red and no traffic then you might go

4. Symbolic NLP

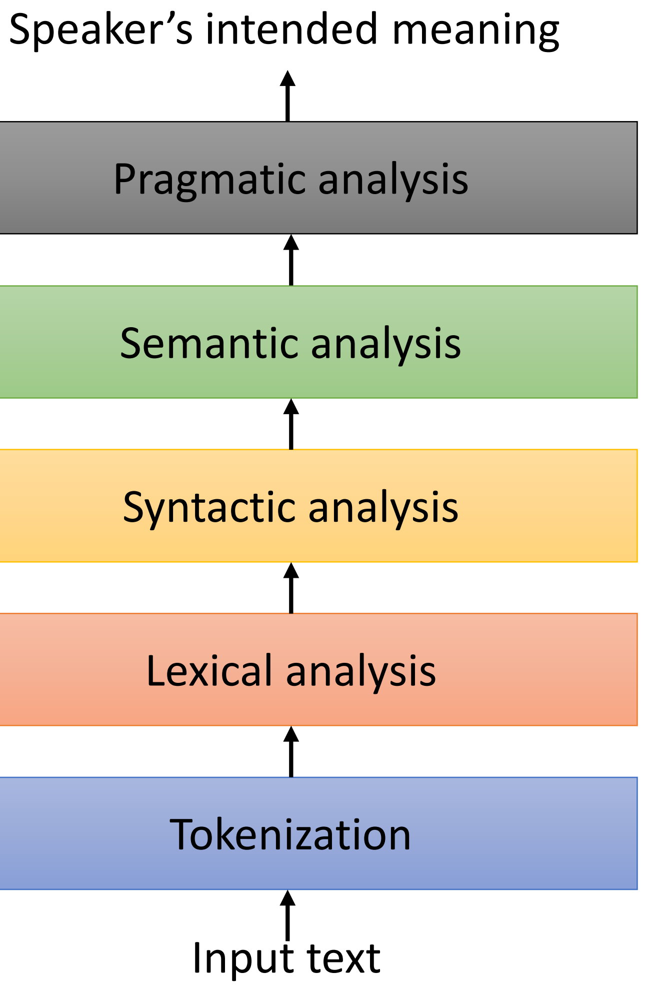
- Worst enemy in NLP: **ambiguity**


HI & LOIS
HOW MUCH DO YOU WANT TO CUT THE GRASS ?
NOT MUCH
DIK BROWNE
© 1986 King Features Syndicate, Inc. World rights reserved
I MEAN HOW MUCH MONEY?
I MEAN I DON'T WANT TO CUT IT MUCH!
IT'S A HARD, HARD LANGUAGE
8-8

Source: Hi and Lois: © 1986 King Features Syndicate, Inc., World Rights Reserved.

4. Symbolic NLP

- The classical **pipeline** in symbolic NLP includes the following stages:



4. Symbolic NLP - Tokenization

- **Tokenization** is a fundamental step both in traditional (symbolic) or modern (statistical or neural) NLP approaches
- Tokenization is defined as a way of separating a piece of text into smaller units called **tokens** (words, characters, or sub-words)
- The goal of tokenization is to create a **vocabulary** (composed by unique token) that is used in next stages of the NLP pipeline
- In this course, we study tokenization in detail on Unit 2

4. Symbolic NLP - Lexical analysis

- **Lexical analysis** study words with respect to their lexical **meaning**
- For example:

Joe Biden elected president $\xrightarrow{\text{Lexical analysis}}$ $\left\{ \begin{array}{l} \text{NNP (proper noun)} \rightarrow \text{Joe} \\ \text{NNP (proper noun)} \rightarrow \text{Biden} \\ \text{VB (verb)} \rightarrow \text{elect} \\ \text{VBD (verb in pas tense)} \rightarrow \text{VB + ‘ed’} \\ \text{NN (noun, singular)} \rightarrow \text{president} \end{array} \right.$

4. Symbolic NLP - Syntactic analysis

- **Syntactic analysis** is aimed to identify the syntactic **structure** of a sentence

Rules

S → NP VP

VP → V NP

VP → VP PP

V → VB

V → VBD

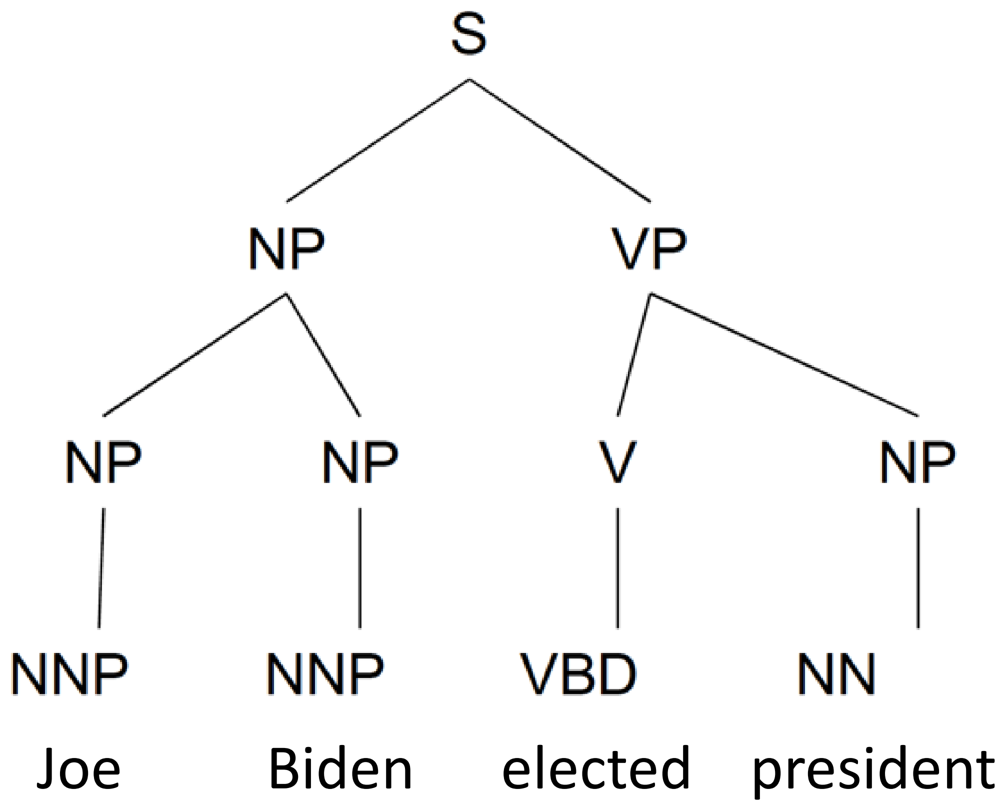
PP → P NP

NP → NP NP

NP → NP PP

NP → NNP

NP → NN



Syntactic categories

S → Sentence

NP → Noun phrase

VP → Verbal phrase

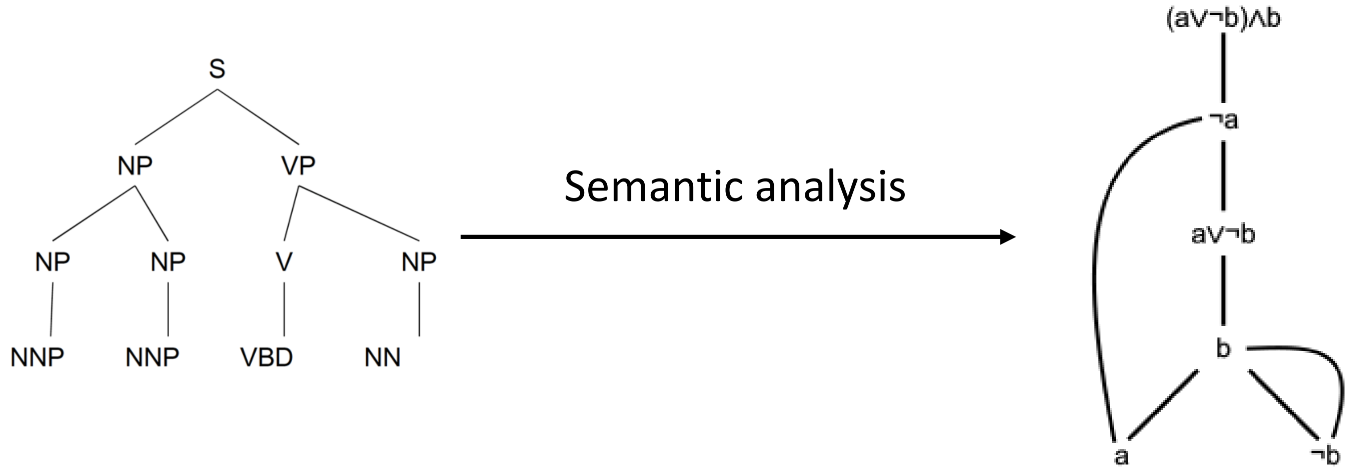
V → Verb

PP → Prepositional phrase

…

4. Symbolic NLP - Semantic analysis

- Given its syntax tree, we can build one or more **semantic representations** in order to capture the real **meaning** of a sentence
- The aim is to build a set-theoretic construction (model) which allows us to create rules of inference (truth)



4. Symbolic NLP - Semantic analysis

- Some **semantic representations** techniques are:

- **Propositional logic** (also known as sentential logic)
  - It is the study in the ways statements can interact with each other
  - It deals with predicates (which can be true or false) and rules of inference between them. For example: $P \rightarrow Q, P \vdash Q$

P implies Q and P is true, therefore Q must be true

- **First-order logic** (also known as predicate logic)
  - It is an extension to propositional logic
  - Propositions are analyzed into predicates and other elements, e.g. quantifiers ($\forall$, $\exists$), constants, variables, or connectives ($\wedge$, $\vee$, $\neg$), among others

- **Lambda calculus** (also written as λ-calculus)
  - It is a Turing complete language formal system

4. Symbolic NLP - Pragmatic analysis

- Pragmatic analysis deals with outside word knowledge (i.e., the **context**), which means knowledge that is external to the input text
  - Pragmatics analysis that focuses on what was described is reinterpreted by what it actually meant, deriving the various aspects of language that require real world knowledge
- Pragmatics ambiguity occurs when the context of a phrase gives it multiple different interpretations
  - Handling this kind of ambiguity is still an open area of research

Table of contents

5. Python

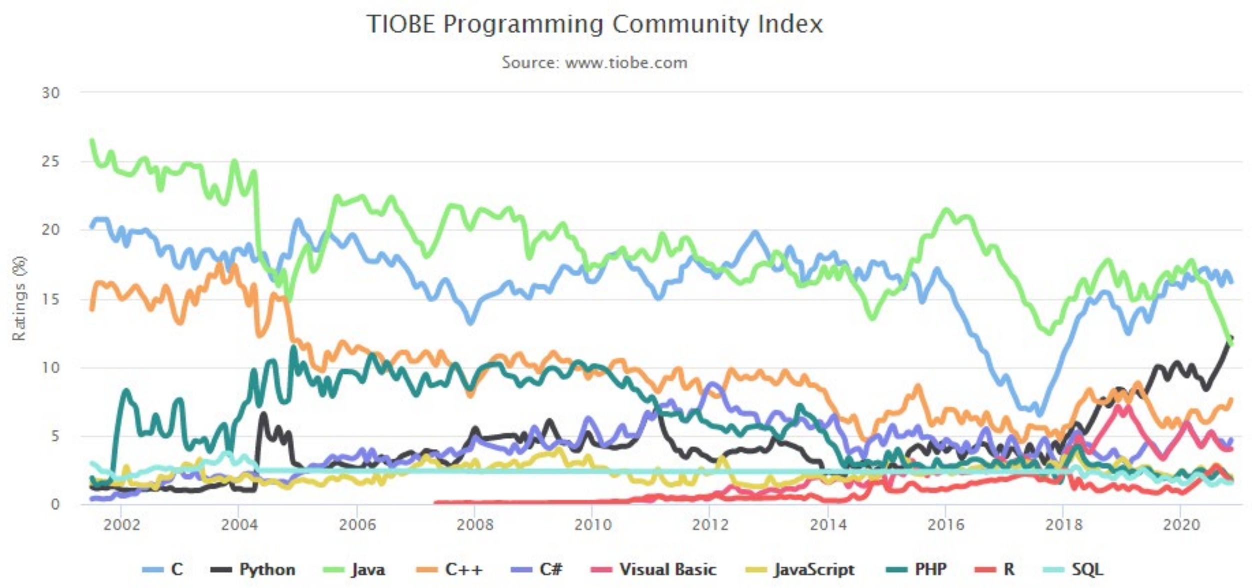
- **Python** is a high-level programming language with a big emphasis on code readability. Its basic features are:
  - Interpreted (Python source code is converted to machine language as long it is executed by the Python interpreter)
  - Cross-platform (Python interpreter is available for major platforms such as Windows, Mac OS, Unix, or Linux)
  - Dynamic typing (the type of each variable is set when a value is assigned)
  - Multi-paradigm (object-oriented together with functional)
- Python version 2 (2.7) was officially discontinued on January 1, 2020. Nowadays, the recommended version for new code in Python is 3.x

python™

https://www.python.org/

5. Python

- The **TIOBE** Programming Community index is an indicator of the popularity of programming languages
- Recently (November 2020), Python has reached 2nd position in TIOBE



https://www.tiobe.com/tiobe-index/

5. Python - Jupyter Notebooks

- **Jupyter** is a nonprofit organization created to “develop open-source software, open-standards, and services for interactive computing across dozens of programming languages”
- Jupyter was created in 2014, as an spin-off project from **IPython**
  - IPython (Interactive Python) is a command shell for interactive programming in multiple programming languages, originally developed for Python
  - Starting with IPython 4.0, the agnostic parts of language moved to an independent project called Jupyter

IP[y]: IPython Interactive Computing

https://ipython.org/

jupyter

https://jupyter.org/

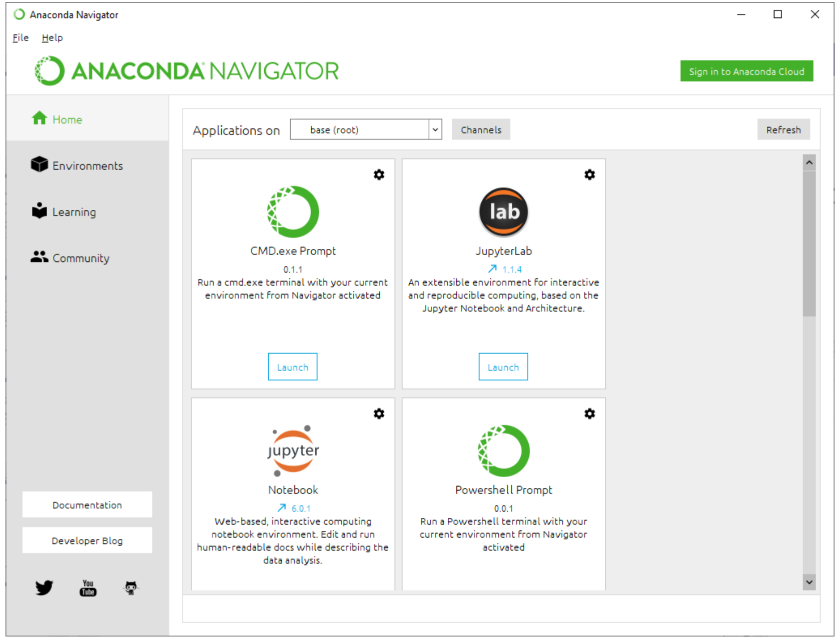
5. Python - Jupyter Notebooks

- **Jupyter Notebooks** (formerly known as IPython Notebooks) are interactive web pages that combine Python code with other elements (images, text, animations, etc.)
- Jupyter notebooks are created in files in extension `.ipynb` and are created/executed in web applications named **Jupyter Notebook Apps**. There are different alternatives to use these apps:

1. Locally. For instance **JupiterLabs**
2. Online. For instance **Google Colaboratory**

5. Python - Jupyter Notebooks

- **JupyterLab** is official web-based user interface for Project Jupyter
  - It can be installed using **Anaconda** (open-source distribution of the Python and R programming languages for scientific computing)



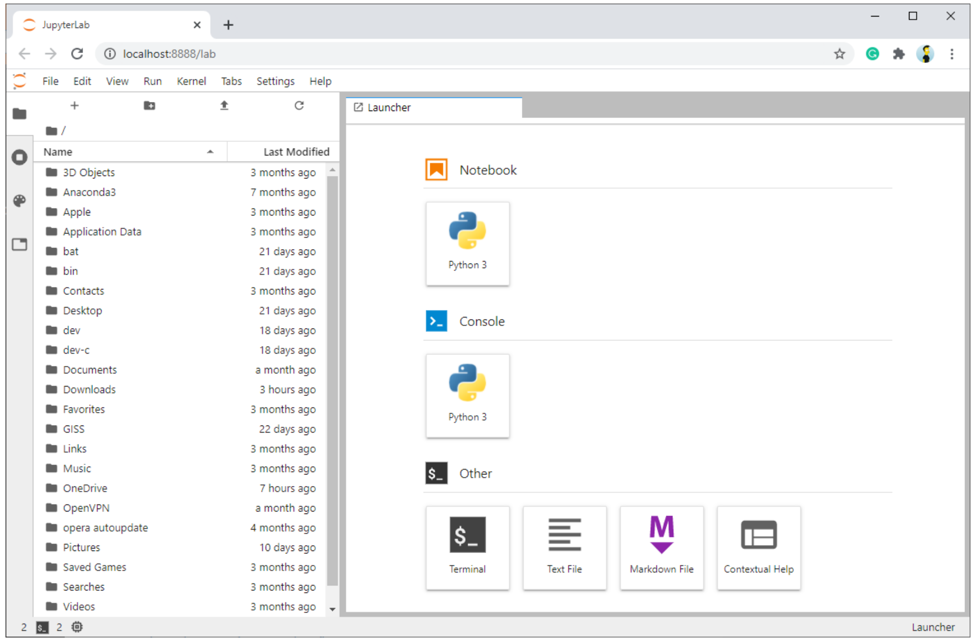
CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated

JupyterLab
1.1.4
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.0.1
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated

https://www.anaconda.com/

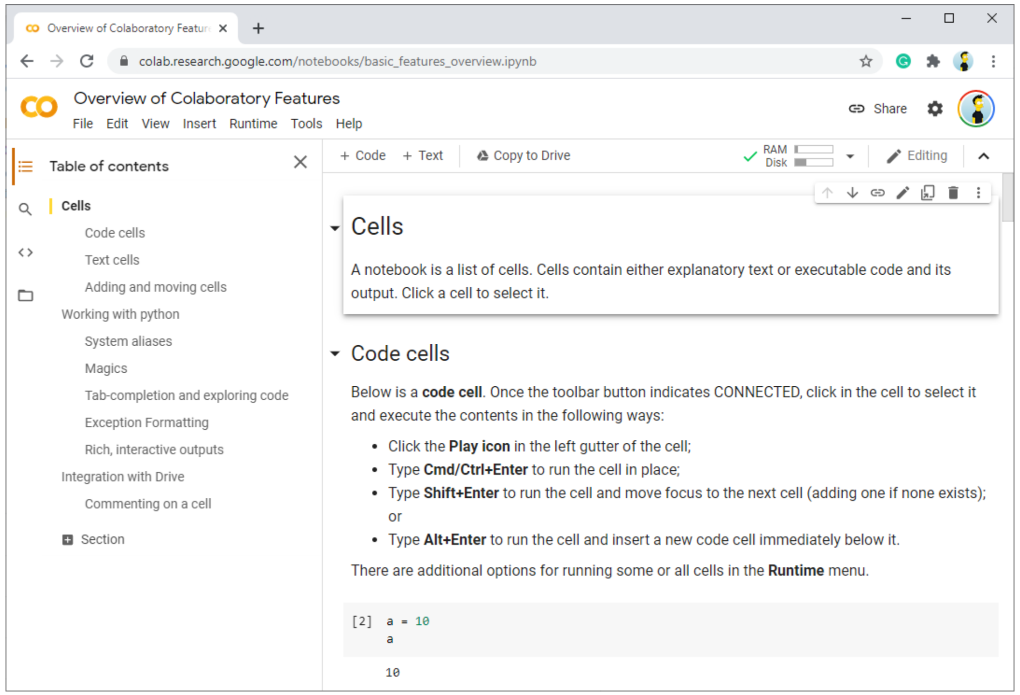


Name	Last Modified
3D Objects	3 months ago
Anaconda3	7 months ago
Apple	3 months ago
Application Data	3 months ago
bat	21 days ago
bin	21 days ago
Contacts	3 months ago
Desktop	21 days ago
dev	18 days ago
dev-c	18 days ago
Documents	a month ago
Downloads	3 hours ago
Favorites	3 months ago
GISS	22 days ago
Links	3 months ago
Music	3 months ago
OneDrive	7 hours ago
OpenVPN	a month ago
opera autoupdate	4 months ago
Pictures	10 days ago
Saved Games	3 months ago
Searches	3 months ago
Videos	3 months ago

https://jupyterlab.readthedocs.io/

5. Python - Jupyter Notebooks

- **Colaboratory** (or simply “Colab”) is an online Jupyter Notebook App created by Google
- It allows to write and execute Python in a web browser
- Main features:
  - Easy to use (zero configuration required)
  - Free access to GPUs (Graphics Processing Unit)
  - Easy sharing
  - Seamless integration with GitHub


Overview of Colaboratory Featur
colab.research.google.com/notebooks/basic_features_overview.ipynb
Share
File Edit View Insert Runtime Tools Help
+ Code + Text Copy to Drive
RAM Disk
Editing

Overview of Colaboratory Features

Table of contents

- Cells
  - Code cells
  - Text cells
  - Adding and moving cells
- Working with python
  - System aliases
  - Magics
  - Tab-completion and exploring code
  - Exception Formatting
  - Rich, interactive outputs
- Integration with Drive
  - Commenting on a cell
- Section

Cells

A notebook is a list of cells. Cells contain either explanatory text or executable code and its output. Click a cell to select it.

Code cells

Below is a **code cell**. Once the toolbar button indicates CONNECTED, click in the cell to select it and execute the contents in the following ways:

- Click the **Play icon** in the left gutter of the cell;
- Type **Cmd/Ctrl+Enter** to run the cell in place;
- Type **Shift+Enter** to run the cell and move focus to the next cell (adding one if none exists); or
- Type **Alt+Enter** to run the cell and insert a new code cell immediately below it.

There are additional options for running some or all cells in the **Runtime** menu.

```
[2]  a = 10
     a

     10
```

https://colab.research.google.com/

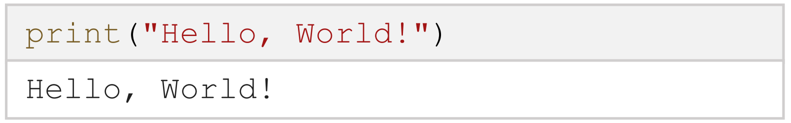
Fork me on GitHub

5. Python - Examples

- All the code examples we see in this module is contained in different Jupyter Notebooks public available on GitHub
  - These examples can be easily imported in Google Colaboratory

https://github.com/bonigarcia/nlp-examples

- For example:



```
print("Hello, World!")
```

```
Hello, World!
```

5. Python - Libraries

- In this course, we use **Natural Language Toolkit** (**NLTK**) as the foundation Python library for NLP
- NLTK can be used both for symbolic and statistical NLP
  - Originally developed by Steven Bird and Edward Loper at the University of Pennsylvania
  - NLTK requires Python version 3.5, 3.6, 3.7, or 3.8
  - Documentation: http://www.nltk.org/
  - Source code: https://github.com/nltk/nltk
  - NLTK supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities

5. Python - Libraries

- **spaCy** is an open-source software library for advanced NLP for Python and Cython (superset of Python designed to give C-like performance)
- It provides a provides a capabilities to carry out task commonly used in any NLP project, including:
  - Tokenization
  - Lemmatization
  - Part-Of-Speech (POS) tagging
  - Entity recognition
  - Dependency parsing
  - Sentence recognition
  - Word-to-vector transformations

spaCy

https://spacy.io/

5. Python - Libraries

- **scikit-learn** is a open-source **ML** library for Python
  - It has been designed to interoperate with the Python numerical and scientific libraries NumPy (library to manipulate vectors and arrays) and SciPy (library for scientific computation, including modules linear algebra, integration, interpolation, optimization, etc.)
- Among other features, scikit-learn supports :
  - Classification: Identifying which category an object belongs to
  - Regression: Predicting a continuous-valued attribute associated with an object
  - Clustering: Automatic grouping of similar objects into sets

https://scikit-learn.org/

5. Python - Libraries

- **Pandas** is an open-source Python library for data manipulation and analysis
  - It has been written as an extension of NumPy
- Pandas is used in the field of Big Data and ML using DataFrames:
  - A DataFrame is a 2-dimensional array used to handle structured data
  - DataFrames allow data manipulation through different operations, such as groupby, join, merge, or melt, among others

pandas

https://pandas.pydata.org/

5. Python - Libraries

- **Beautiful Soup** is a Python package for parsing HTML and XML documents
  - It creates a tree for parsed pages that can be used for **web scraping** (i.e., extract data from web pages)

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

5. Python - Libraries

- **Keras** is an open-source Python library that provides a user-friendly API for ANN
  - It is optimized for common use cases which provides clear and actionable feedback for user errors
- Keras acts as an interface for **TensorFlow**
  - TensorFlow is an open-source library for ML developed at Google
  - It provides high-level and low-level APIs in different languages, such as Python, JavaScript, C++, Java, Go, or Swift

Keras

https://keras.io/

TensorFlow

https://www.tensorflow.org/

Table of contents

6. References

- Symbolic NLP:
  - Indurkhya, N., & Damerau, F. J. (2010). ***Handbook of Natural Language Processing (2nd edition)***. CRC Press.
- Symbolic and statistical NLP:
  - Bird, S., Klein, E., Loper, E. (2009) ***Natural Language Processing with Python***. O'Reilly Media. http://www.nltk.org/book/
- Statistical and neural NLP:
  - Arumugam, R., Shanmugamani, R. (2018). ***Hands-On Natural Language Processing with Python***. Packt Publishing.
  - Hapke, H.M, Hobson, L., Howard, C. ***Natural Language Processing in Action***. (2019). Manning.
  - Kedia, A., Rasu, M. (2020). ***Hands-On Python Natural Language Processing***. Packt Publishing.
- Neural NLP:
  - Goyal, P., Pandey, S., Jain, K. (2018). ***Deep Learning for Natural Language Processing***. Apress.
  - Singh, P., Manure, A. (2020). ***Learn TensorFlow 2.0***. Apress.
- Machine learning :
  - Joshi, P. (2016). ***Python Machine Learning Cookbook***. Packt Publishing.

Table of contents

7. Module planning

- Module 5 (NLP) of this course is made up by 4 units:

Unit	Schedule	Practice
1. Introduction to NLP	1 session: 20/11	-
2. Datasets for NLP	2 sessions: 23/11 and 27/11	1. Vocabulary build
3. Statistical NLP	2 sessions: 30/11 and 4/12	2. Text classification
4. Neural NLP	2 sessions: 11/12 and 14/12	3. Sentiment analysis
Review	1 session: 18/12	-

- All classes are **online** (Mondays 15:15-16:45 and Fridays 18:00-19:30)
- Module 5 counts the **20%** of the final score of the course
  - It will be evaluated with the abovementioned 3 practices (to be solved **individually**)
  - The solution should be submitted as a Jupyter Notebook in Aula Global
  - Each practice has the same weight (**1/3** each) in the final module score

Table of contents

8. Takeaways

- **Natural Language Processing** (**NLP**) is an interdisciplinary discipline concerned with the manipulation and understanding of natural languages (such as English, Spanish, etc.)
- The main approaches for NLP are: symbolic (classical, based on grammar rules), **statistical** (based on ML), and **neural** (based on ANNs and DL)
- In this module we will use **Python** to create **Jupyter Notebooks** which implements basic NLP applications (e.g. text classification) using statistical and neural techniques
- For the practices, we will use different Python libraries, such as **NLTK** (e.g. for tokenization or stemming), **Beautiful Soup** (for web scrapping), **scikit-learn** (for ML), and **Keras** (for ANN and DL)