Radboud University

Assignment 2: Information Extraction

Extracting timelines and matching events from biographies

LET-REMA-LCEX06-2020 Text and Multimedia Mining

After this assignment:

- You become familiar with the intricacies and challenges of time expression labelling and finding matching events.
- You can define a set of patterns for extracting time patterns from text.
- You can calculate precision for the output of your code on an unseen text.
- You gain insight into the importance of pattern generalizability.

Extracting a timeline from a biography

In this exercise, you will create timelines of events described in two texts and you will find the matching events (i.e. overlapping dates) of the two timelines. Please use the Python programming language. In exceptional cases, it is allowed to use another programming language. Please contact the TAs to request an exception.

- Examine pierre.txt, the biography of **Pierre Curie**, and notice the patterns in which the date expressions occur (do not yet look at marie.txt, an extraction of the biography of Marie Curie). A data expression is a sequence within the text (can contain letters, numbers, and/or punctuation) that expresses a point in time or a period of time.
- Write a function that:
  - extracts date expressions from a text.
  - converts the dates to the ISO 8601 date standard and *orders* the dates chronologically (more info and examples: https://www.iso.org/iso-8601-date-and-time-format.html)
  - prints a table with in the left column the standardized dates and in the right column for each date the sentence from which the date was extracted.
- Repeat until satisfied:
  - Run your function on pierre.txt.
  - Make adaptations to your code if necessary.
  - Go through the output manually and calculate precision.
- After being satisfied with your patterns, calculate precision. You need this precision for the report.
- Run your script on the marie.txt, the yet unseen biography of **Marie Curie**.
- Go through the output manually and calculate precision. You need this precision for the report.
- Make adaptations to your code if necessary. If you adapt the code, calculate final precision for the report.
- Find the matching events (i.e. overlapping dates in the two biographical time lines) between the two timelines. We highly encourage you to use the 'comm' function on your ISO-8601 dates in your command line (more info and example: https://www.computerhope.com/unix/ucomm.htm. Note that

this command also works on Mac OS/Windows). It is also allowed to find the matching events programmatically in Python or by using an (online) tool (for example: https://text-compare.com/). Do not search for matching events manually, i.e. by hand.

- Write 1.5-page report in which you
  - Show the **timeline** of the life of Marie Curie in a table format: the left column showing the date in ISO-8601 format and the right column showing the sentence containing the date. Make sure the ordering of the dates is correct.
  - Discuss the difficulties you encountered during the development of the time patterns and extracting the timeline. Also mention the adaptations that you needed to make for processing the unseen biography marie.txt.
  - Show the list of **matching events** that you found, in ISO-8601 format.
  - Discuss the list of matching events that you found. When going through the texts manually, do you find matching events that you did not find programmatically/automatically? If so, what could be the reason(s) for this? Discuss your answers to these questions and any other difficulties you encountered during the extraction of matching events.
  - Report the **precision** of your date expression extractor for both pierre.txt and running *the same code* on marie.txt. If more adaptations were made afterwards, do also report on the final precision as well.
  - Compare the precisions of pierre.txt and marie.txt. Are there any differences? If so, where does this difference come from? What does this difference mean in terms of generalizability? Discuss your answers to these questions in the report.

Some hints:

- Check out https://regexr.com/ for testing and refining the regular expressions you use to capture date expressions. It also has a handy cheat sheet you can use.
  For coding newbies: You can contact the TAs to get a regex example function in Python.
- For converting strings into ISO-compliant dates, check out Python's inbuilt datetime module. With a quick Google search, you'll find functions in there that can make this task a lot easier.
- You are allowed to use NLTK (or another NLP toolkit) for processing the text.
- Please try to write your own patterns for time extraction and do not rely on libraries that automatically extract date expressions as learning about regular expressions is one of the learning objectives of the exercise.

Submission instructions

- Convert your report to a .pdf file.
- Upload it in the folder named 'A2: Information Extraction'. Please name it txmm_a2_studentnumber.pdf
- Upload your code to the folder named 'A2: Information Extraction (Only code)'. If you have multiple files, please zip them beforehand as you can only submit a single file.

Note that the report is assessed. The code is not assessed, but you need to submit it to pass. We collect the code to be able to get more insight into the approaches that you are taking and to give you feedback concerning readability.