








COMP 474/6741 Intelligent Systems (Winter 2022)

Worksheet #8: NLP & Text Mining

-  **Task 1.** Text Mining systems are about extracting interesting (and possibly actionable) knowledge from unstructured, natural language documents. To see an example in action, test out the *Europe Media Monitor (EMM)* system, which analyses thousands of news sources in over 70 languages and extracts information about events, people involved, and what they said: <https://emm.newsbrief.eu/> Find a “top story”, look at the people involved and for a person, look at the “*Extracted quotes from*” and “*Extracted quotes about*”.
-  **Task 2.** A major framework for developing text mining applications is the open source *General Architecture for Text Engineering (GATE)*, <https://gate.ac.uk/>. The “hello world” type example application for GATE is ANNIE, a *Nearly-New Information Extraction System*, which detects a number of so-called *Named Entities (NEs)* in a text: Try it out with the online demo at <http://services.gate.ac.uk/annie/>.
-  **Task 3.** In this course, we will work with the *spaCy* NLP library for Python. The first step in an NLP pipeline is *Tokenization*, splitting an input stream of characters into individual tokens. See how it works either with your own installation of spaCy or the online code demo at <https://spacy.io/usage/spacy-101#annotations>.
-  **Task 4.** Sentence splitting (a.k.a. sentence segmentation or sentence boundary detection) is another elemental but crucial step in text processing. Look at the alternative methods available in spaCy and understand how to access the sentences in a document object after processing at <https://spacy.io/usage/linguistic-features#sbd>.
-  **Task 5.** Assigning *part-of-speech* (POS) tags to each token is another essential step in NLP. Process a document and see how to access the POS tags from `token.pos_` at <https://spacy.io/usage/linguistic-features#pos-tagging>. Look at the definitions for some of these tags using the references mentioned in the lecture.
-  **Task 6.** Moving on the syntactic analysis of a sentence, spaCy has a *dependency parser* built into the default pipeline. Try it on some text at <https://spacy.io/usage/linguistic-features#dependency-parse> and understand how the *dependencies* model the syntactic structure of a sentence.
-  **Task 7.** Similar to GATE, spaCy includes a pre-trained model for *Named Entity Recognition (NER)*. Process some text and see what entities it recognizes at <https://spacy.io/usage/linguistic-features#named-entities>. You can also see an online visualization using displayCy at <https://explosion.ai/demos/displacy-ent>.
-  **Task 8.** Now let’s add the recognition of some new entities using rules that run over different linguistic features, like tokens and their POS tags. Write rules that detect entities of type *University* in a document, like “*Concordia University*” or “*University of Waterloo*”. Write a few rules that mix different linguistic features and run it using the online interface at <https://explosion.ai/demos/matcher>. *Note:* the online demo doesn’t show you overlapping matches, you’d have to work with these in Python code.