The Natural Language Processing pipeline

For Technical Competences

Team semester 5

June, 2024

Abstract

This document contains a brief introduction to NLP pipelines.

Contents

1
Preprocessing steps 2
Advanced NLP models 2
Utilizing these modules 2
Popular (python-based) NLP tools 2
Integrating NLP tools 3
Other 3

Text data is inherently unstructured. Extracting valuable information from text typically requires converting this unstructured data into a more structured format. This transformation is achieved through Natural Language Processing (NLP) pipelines, which consist of several interconnected modules. The most common modules include:

1. **tokenizer**: Splits text into paragraphs, sentences, and words.
2. **part-of-speech (POS) tagger**: Identifies parts of speech (verbs, nouns, etc.).
3. **lemmatizer**: Maps words to their base forms (lemmas).
4. **named entity recognizer (NER)**: Detects entities such as people, locations, and organizations.
5. **dependency parser**: Analyzes sentence structure.
6. **semantic role labeler (SRL)**: Determines the roles of entities within a sentence (e.g., who does what to whom).
7. **word sense disambiguation (WSD)**: Assigns the correct meaning to each word.
8. **polarity tagger**: Determines the sentiment (positive or negative) of a sentence.

Not all modules are necessary for every task, but it's useful to be aware of their existence for when specific needs arise.

Preprocessing steps

Before applying these modules, several preprocessing steps are often necessary:

- **lowercasing**: Converting all text to lowercase to ensure uniformity.
- **removing stop words**: Eliminating common words (e.g., "the", "and") that typically do not contribute to the meaning.
- **stemming**: Reducing words to their base or root form.

Advanced NLP models

Modern NLP tasks increasingly leverage advanced neural network models:

- **BERT**: Bidirectional Encoder Representations from Transformers, useful for understanding the context of words in a sentence.
- **GPT**: Generative Pre-trained Transformer, excellent for text generation tasks.
-
- **LLaMA**: Large Language Model Meta AI, a Meta competitive from OPENAI's GPT.
- **Transformers**: A broad class of models that have revolutionized NLP with their attention mechanisms.

Utilizing these modules

It's important to note that Python is not the only option for working with these modules. There are many robust NLP programs that can be operated via the command line. Here are a few examples:

1. **TreeTagger**: A combined POS-tagger and lemmatizer supporting many languages. For Python integration, you can use `treetaggerwrapper` or the slower `Treetagger-python`.
2. **Stanford's CoreNLP**: A powerful system capable of processing multiple languages including English, German, Spanish, French, Chinese, and Arabic. The English pipeline is the most comprehensive.
3. **MaltParser**: Offers models for English, Swedish, French, and Spanish. Many more tools are available, and you'll explore these further in the NLP toolkits course. Often, it's efficient to run these programs directly and then analyze their output using Python.

Popular (python-based) NLP tools

Several NLP tools have been specifically developed for Python, offering a range of functionalities:

1. **Natural Language Toolkit (NLTK)**: A versatile library with a broad range of features, though it may not be the fastest or most state-of-the-art.
    - Supports corpus analyses and interfaces with WordNet.
    - Access to numerous corpora.
    - Ability to create POS-taggers, with state-of-the-art performance if sufficient training data is available.
2. **SpaCy**: Offers a tokenizer, POS-tagger, parser, and entity classifier for English and German, with more languages in progress.
3. **Pattern**: Describes itself as a 'web mining module', providing a tokenizer, tagger, parser, and sentiment analyzer for multiple languages, along with APIs for Google, Twitter, Wikipedia, and Bing.
4. **TextBlob**: A general NLP library built on NLTK and Pattern.
5. **Hugging face transformers**: Provides state-of-the-art general-purpose architectures for NLP tasks.
6. **AllenNLP**: An open-source NLP research library built on PyTorch

Integrating NLP tools

To effectively integrate different NLP modules or tools within a single project:

- Use APIs and wrappers to bridge different tools and languages.
- Ensure compatibility of data formats between different tools.
- Use intermediate storage (e.g., JSON files) for passing data between different stages of the pipeline.

Other

- **python-ftfy**: A library that fixes common Unicode errors, making text readable and properly formatted.
- **Networkx**: One of the best libraries for working with data represented as graphs. It supports various graph algorithms and can export to Gephi format for creating beautiful visualizations.
- **Flask**: A lightweight framework for creating websites and web applications. It is particularly useful for developing demos or browser-based interfaces quickly and easily.
- **Django**: A more powerful and comprehensive framework for creating robust websites and web applications. It includes numerous built-in features for handling tasks such as authentication, database interaction, and templating.
- **SortedContainers**: Provides containers that maintain their elements in sorted order, offering fast and efficient sorting operations

---

T. Busker, Aug. 2025