Introduction to NLP

Iván Moreno (ivan@nieveconsulting.com)

Table of Contents

Introduction

- **Natural Language Processing (NLP)** is a subfield of artificial intelligence that focuses on the interaction between computers and humans using natural language.
- NLP enables computers to understand, interpret, and generate human language, allowing for more natural communication between humans and machines.
- In this presentation, we will explore the fundamentals of NLP, including key concepts, tasks, and approaches.

NLP Tasks

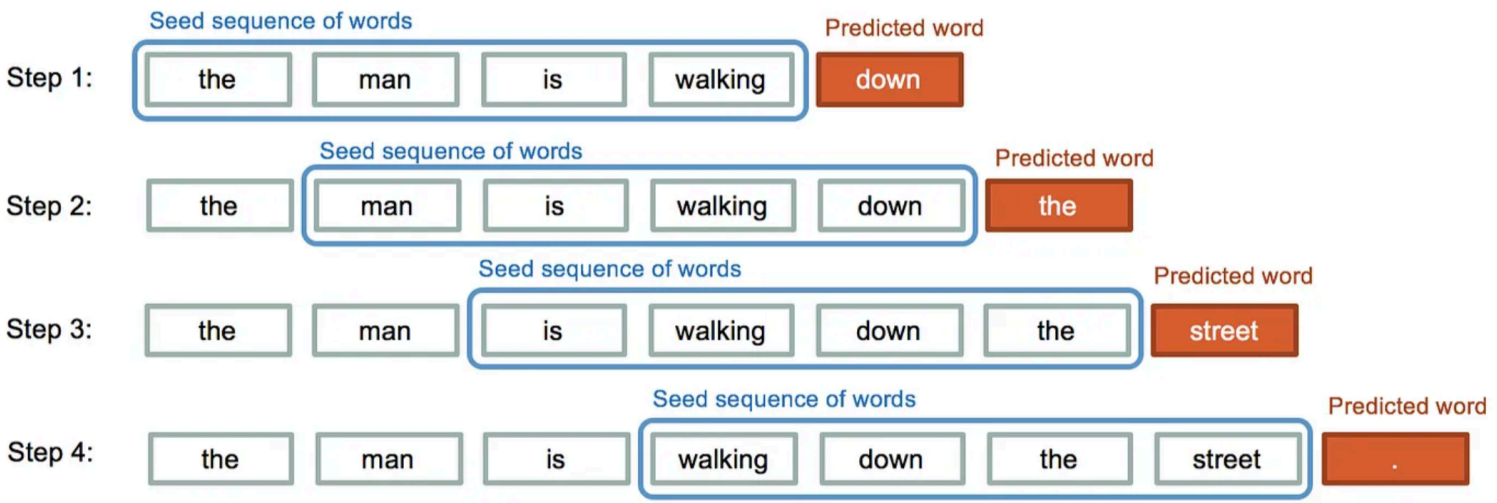
- NLP tasks involve processing and understanding text data to perform specific functions.
- Common NLP tasks include *sentiment analysis*, *named entity recognition*, *machine translation*, and more.

- **Text Classification**: Assigning categories or labels to text data. *Allows for organizing, filtering and analyzing text.*
- **Named Entity Recognition**: Identifying entities like names, locations, and organizations in text. *Allows for extracting structured information from unstructured text.*
- **Machine Translation**: Translating text from one language to another.
- **Part-of-Speech Tagging**: Assigning grammatical tags to words in a sentence. *Fundamental for understanding grammatical structure of sentences, enabling downstream tasks like parsing and entity recognition.*
- **Sentiment Analysis**: Determining the sentiment or emotion expressed in text. *Widely used for social media monitoring, customer feedback analysis, and more.*
- **Question Answering**: Generating answers to questions based on text data. *Used in chatbots, search engines, and more.*
- **Text Summarization**: Creating concise summaries of longer text documents. *Enables digestion of large volumes of text.*
- **Language Modeling**: Predicting the next word in a sequence of text. *Fundamental for many NLP tasks like machine translation, text generation, and more.*
- and more…

Understanding Text Data

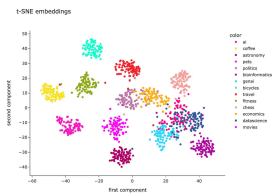
- Text data is a common form of unstructured data that is prevalent in various domains.
- There exist several ways of treating text data. Depending on the approach, a different set of models and techniques can be used.
  - **Sequence Data**: text is most commonly treated as a sequence of tokens (e.g., words, characters, or subwords). The sequential nature of text is important for capturing context, grammar, and meaning.
    - Use cases: sentiment analysis, named entity recognition, machine translation.
    - Models: RNNs.
  - **Bag-of-Words**: representing text as a collection of words, ignoring word order.
    - Use cases: text classification, topic modeling.
    - Models: Naive Bayes, SVM.
  - **Graphs**: Representing text as a network of words and their relationships, with words / sentences as nodes and relationships as edges.
    - Use cases: semantic analysis, knowledge graphs.
    - Models: Graph neural networks.
  - **Sets**: Treating text as a set of features, focusing on properties of the text rather than the order of words.

Text Data as Sequences



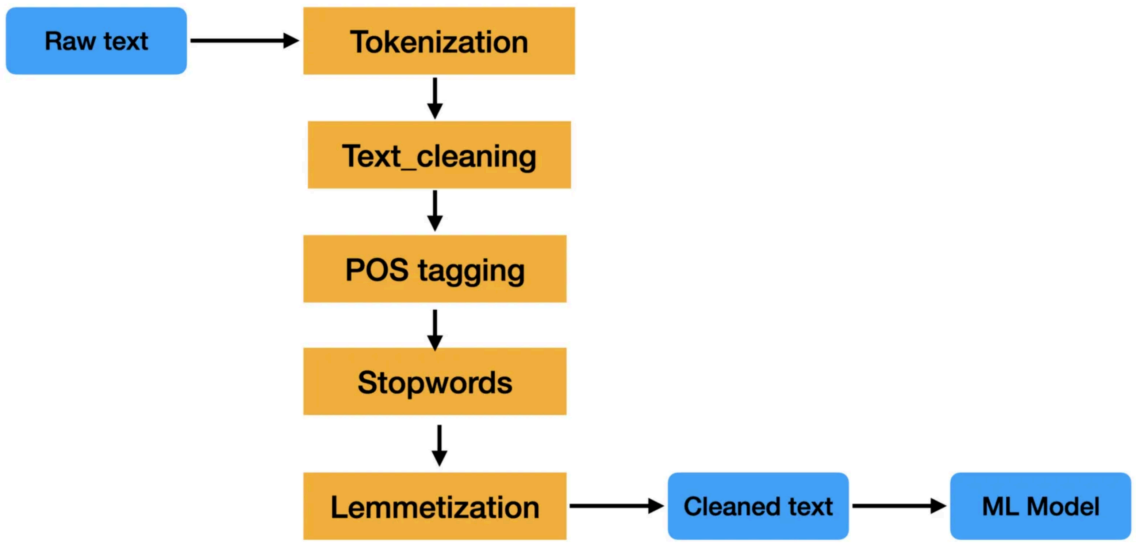
- Text data is inherently sequential, with words forming a sequence that conveys meaning.
- We can represent text data as sequences of tokens (words, characters, subwords) to capture context and relationships.
- In this representation, each token in the sequence depends on the tokens that precede / follow it, reflecting the natural structure of language.
- However, sequential models can struggle with long-range dependencies, and processing text strictly sequentially is inefficient.

Text Data as Embeddings



- Text data can be represented as embeddings, which are dense vectors that capture semantic meaning.
- Embeddings map discrete data (words, sentences) to continuous vector spaces where semantic relationships are preserved.
- The resulting learned vector space captures relationships between words, such as synonymy or semantic gradients (e.g., continuum from “bad” to “good”).
- There exist two main types of embeddings: static embeddings (pre-trained, fixed) and contextual embeddings.
  - Contextual embeddings change based on the context in which a word appears, capturing nuances and meaning shifts. This is helpful for cases where word meaning depends on the context (e.g., polysemy).

NLP Pipeline



- In order to perform NLP tasks, *text data goes through a series of processing steps*.
- The NLP pipeline consists of tasks like **tokenization**, **part-of-speech tagging**, **named entity recognition**, and more.
- Each step in the pipeline adds structure and meaning to the text data, enabling more complex analyses and applications.
- Upon processing, the text data is then suitable for modeling, analysis, or other tasks.

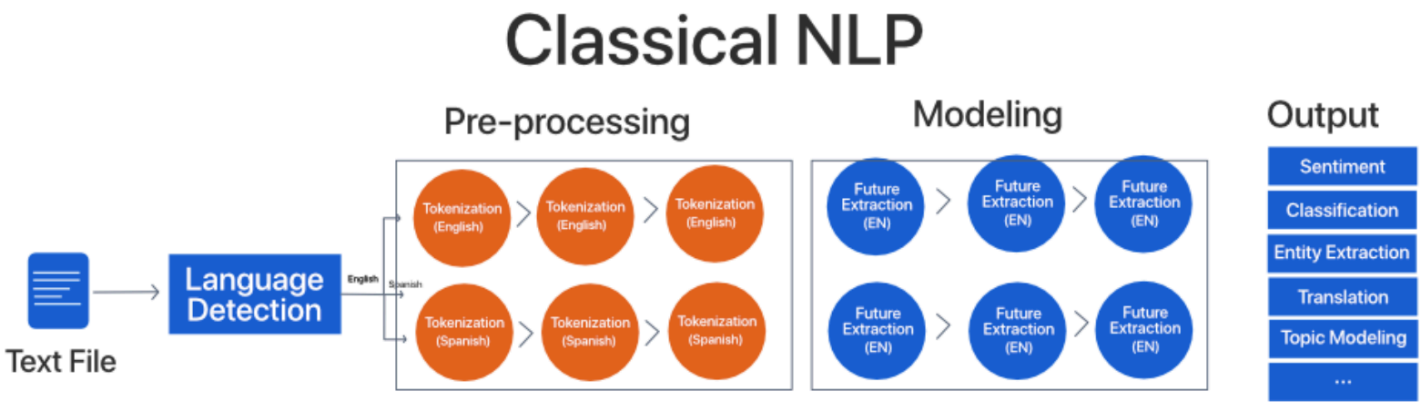
Preprocessing

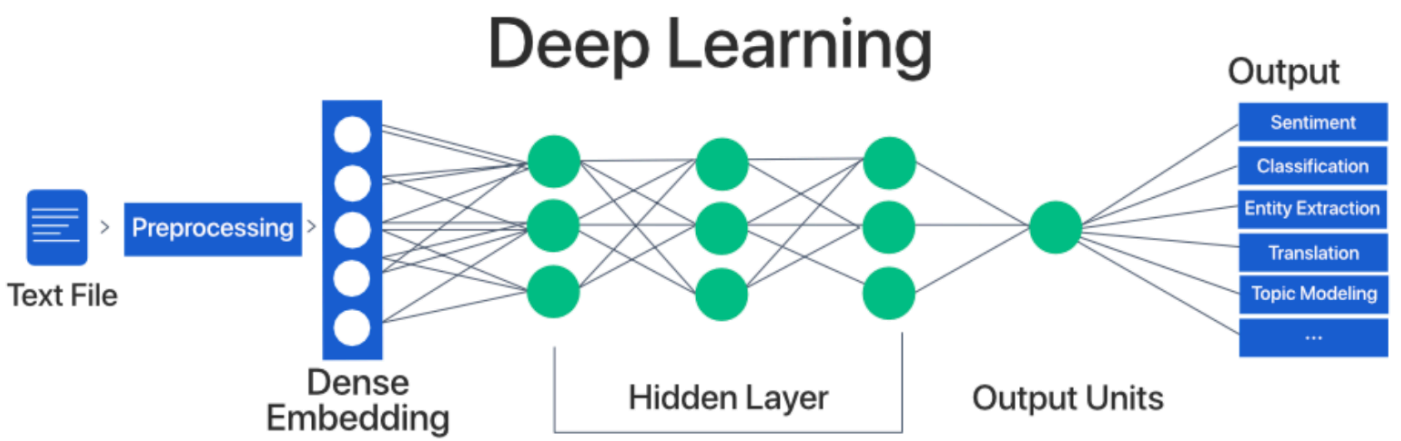
- Steps taken to prepare text data for analysis. The goal is to remove noise, standardize text, and ensure consistency.
- **Tokenization**: Breaking text into individual words or subword units. For example, `I love NLP` becomes `[I, love, NLP]`.
- **Lowercasing**: Converting text to lowercase, ensuring uniformity. For example, `Hello` becomes `hello`.
- **Stemming and Lemmatization**: Reducing words to their base or root form. For example, “running” becomes “run.”
- **Sentence Segmentation**: Splitting text into sentences. For example, `This is a sentence. This is another sentence.` becomes `["This is a sentence.", "This is another sentence."]`.
- **Stopword Removal**: Eliminating common words that carry little meaning. Examples include “the,” “is,” and “and.”
- **Normalization**: Ensuring consistent spelling, punctuation, and formatting. For example, converting “US” to “United States.”

Feature Extraction

- **Feature extraction** involves converting text data into numerical or categorical features that machine learning models can understand.
- **Bag-of-Words**: Representing text as a vector of word counts or frequencies, ignoring word order.
- **TF-IDF**: Assigning weights to words based on their frequency in a document and across the corpus. The frequency is used as a proxy for relevance.
- **N-grams**: Capturing local context by considering sequences of words. For example, bigrams (2-grams) and trigrams (3-grams) capture pairs and triplets of words, respectively.
- **Word Embeddings**: Mapping words to dense vectors that capture semantic meaning.
  - **Word2Vec**, **GloVe**, **FastText**, **BERT**.

Modeling Approaches





- Depending on the task, different modeling approaches can be used to process text data.
- **Rule-Based Systems**: Define rules to process text. Useful for simple tasks like keyword matching.
- **Statistical and Machine Learning Models**: Use statistical techniques to model text data. Common models include Hidden Markov Models, SVMs, and Naive Bayes.
- **Deep Learning Models**: Employ neural networks to process text. Common deep learning models include RNNs, CNNs, and Transformers.