Natural Language Processing (NLP)

Core of Modern NLP

Equipping You with Research Depth and Industry Skills – Data Science Oriented

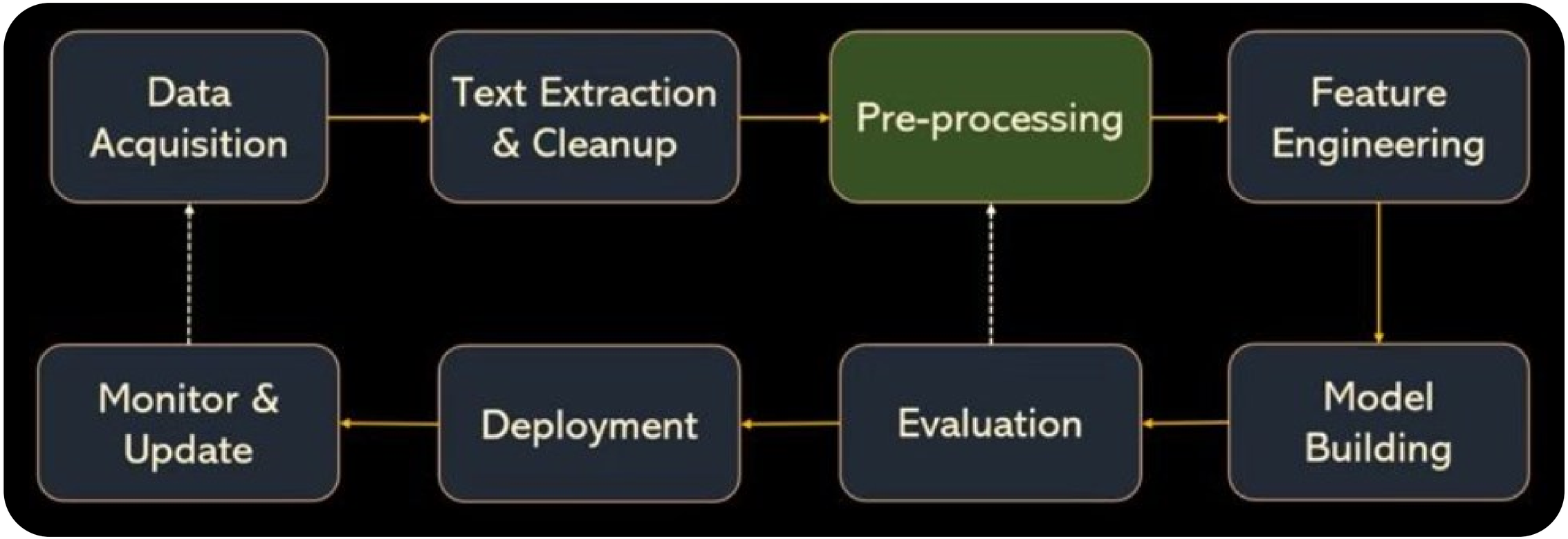
By:

Dr. Zohair Ahmed

www.youtube.com/@ZohairAI Subscribe

www.begindiscovery.com

NLP Pipeline



NLTK: Overview

- Natural Language Toolkit
- Released in **2001** (one of the oldest NLP libraries).
- **Strengths:**
  - Rich in linguistic resources (corpora, lexicons).
  - Many classic algorithms implemented (tokenizers, parsers, stemmers).
  - Great for **teaching, research, prototyping**.
- **Weaknesses:**
  - Slower than SpaCy.
  - Not optimized for production.

NLTK: Overview

- **SpaCy: Overview**
- Released in **2015** (modern NLP library).
- **Strengths:**
  - Industrial-strength, fast, efficient.
  - Built-in support for **deep learning** (integrates with PyTorch, TensorFlow).
  - Pre-trained pipelines for **NER, POS, dependency parsing**.
  - Easy API for production.
- **Weaknesses:**
  - Smaller set of linguistic resources.
  - Less useful for "teaching old-school NLP".

What is Tokenization?

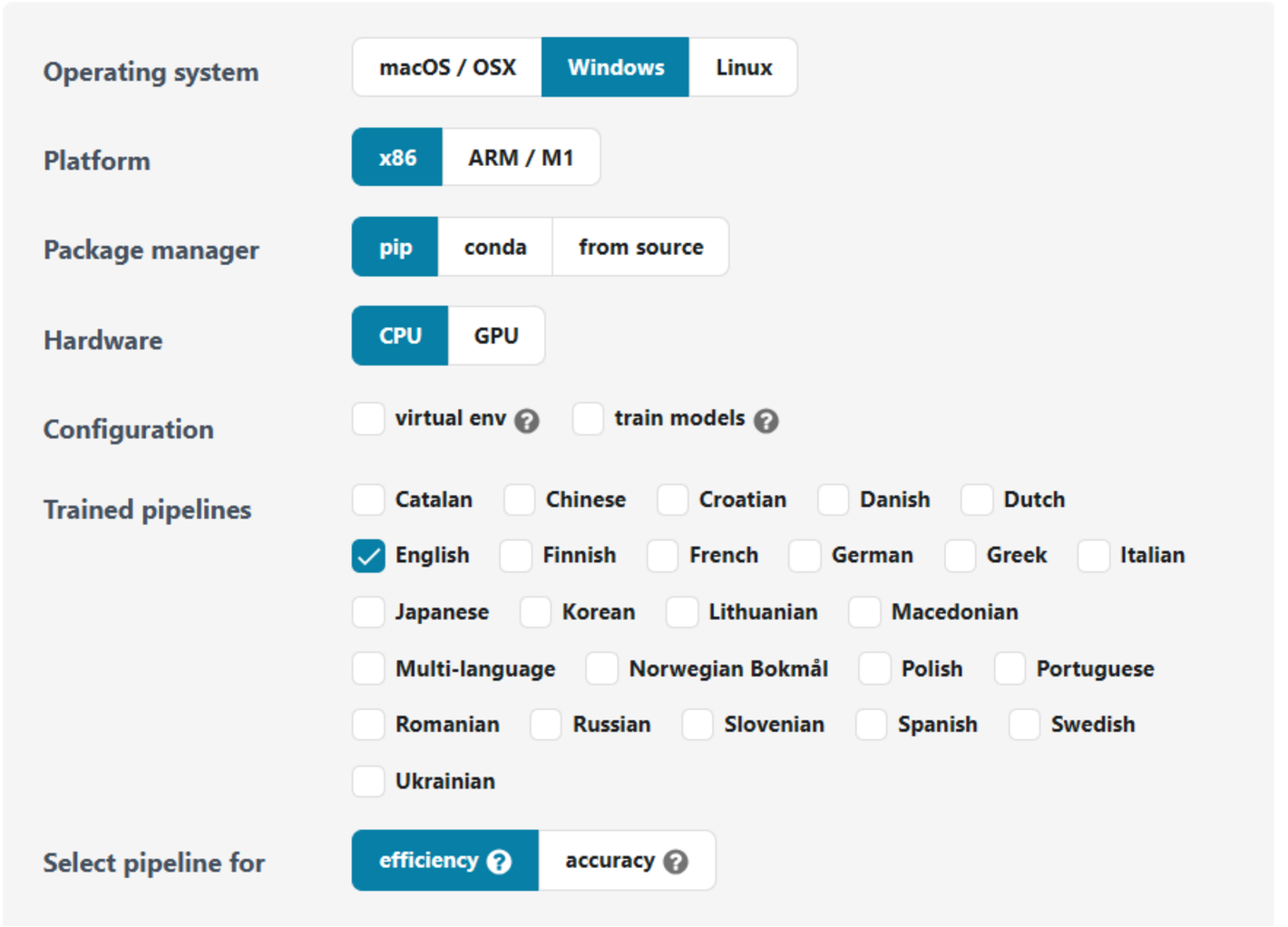
- **Definition:** Splitting text into **meaningful segments** (tokens)
- Types:
  - **Sentence Tokenization** → Paragraph → Sentences
  - **Word Tokenization** → Sentence → Words
- Why not just split by spaces or dots?
  - Ambiguities:
  - "Dr. Strange went to N.Y."
  - "U.S.A. is a country"

Why We Need NLP Libraries?

- Simple rules fail:
  - "Dr." ≠ End of sentence
  - "N.Y." ≠ End of sentence
- Tokenization requires:
  - Language-specific rules
  - Exceptions handling (abbreviations, punctuation, etc.)
- Libraries like spaCy provide robust tokenizers

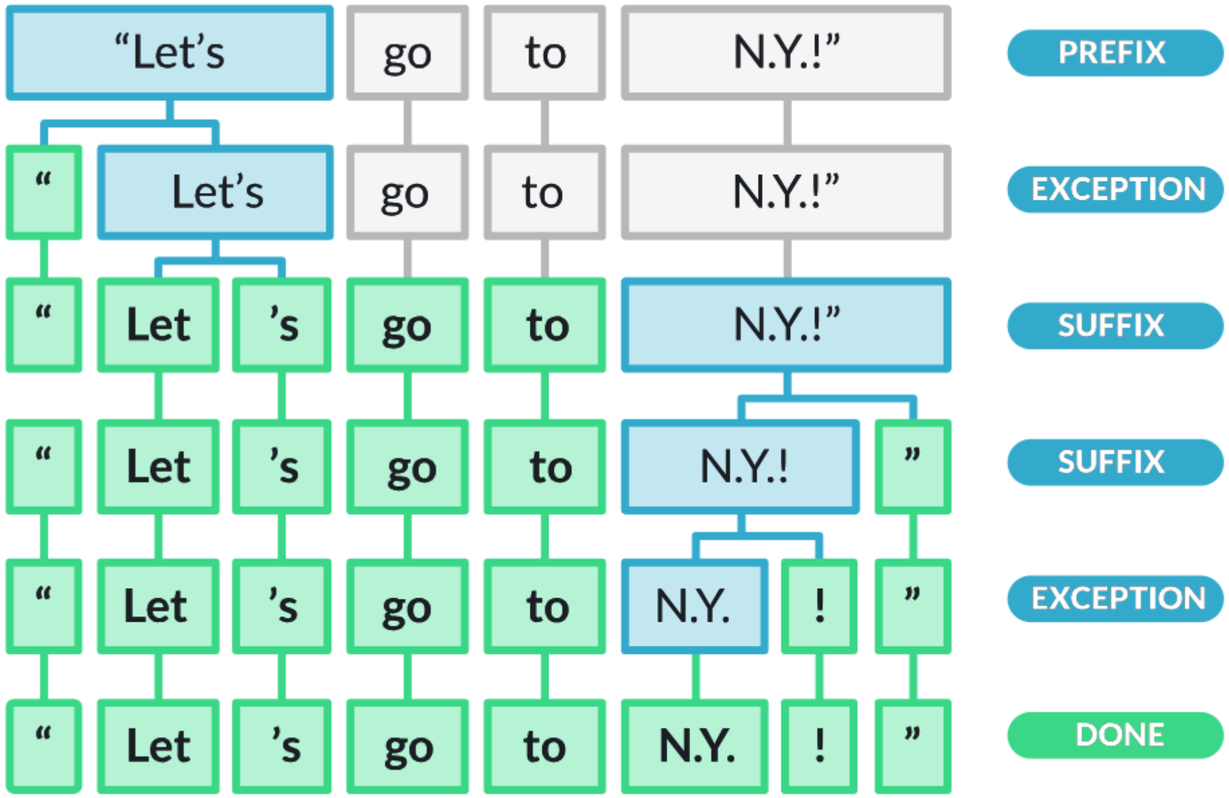
Work with Libraries

- **pip install spacy**
- "en" = English, "de" = German, "fr" = French, "hi" = Hindi
- Blank model = only tokenizer
- Pre-trained model = includes parser, NER, POS tagging (covered later)



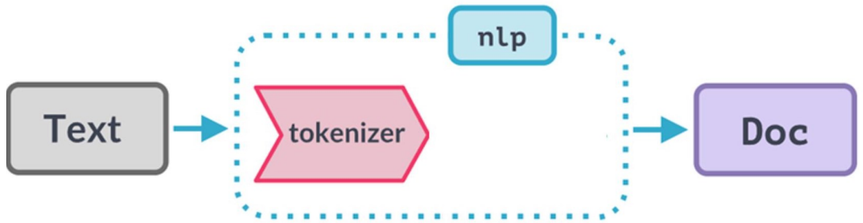
Work with Libraries

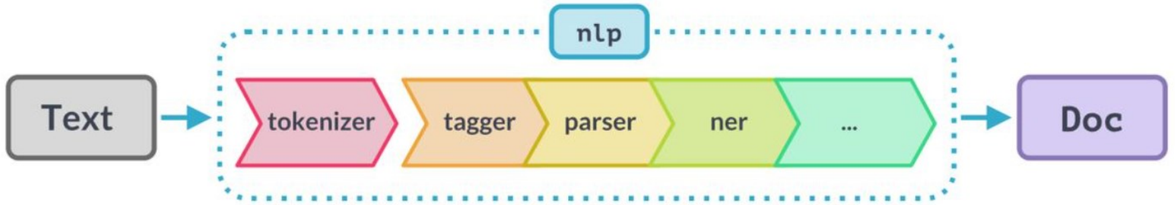
- Split the quotes " " as prefix/suffix
- Split "Let's" into "Let" + "'s" using exception rules
- Kept "N.Y." as one token using abbreviation exception
- Split "!" as suffix



What is a Pipeline?

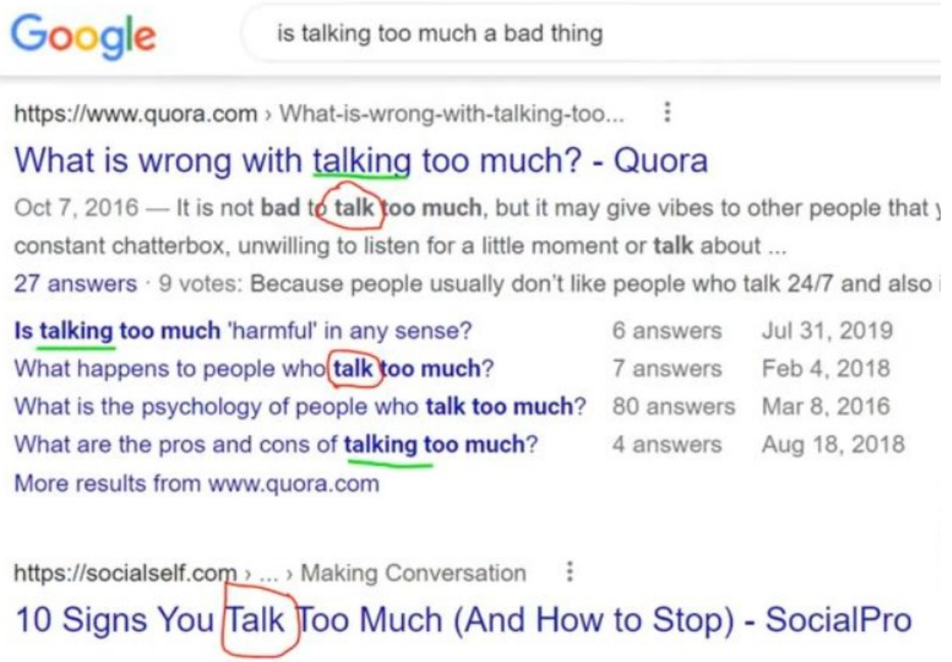
- **Pipeline = sequence of components after tokenizer**
- Input: **Text** → Tokenization → Components → **Doc object**
- Components can include:
  - **Tagger** (Part-of-Speech)
  - **Parser** (Dependencies)
  - **NER** (Named Entity Recognition)
- **Blank vs Complete Pipeline**





Why Reduce Words to Base Form?

- Google search: *“talking”* → also shows results for *“talk”*
- Text classification: *talking, talked, talks* → all mean **talk**
- Helps:
  - Reduce **vocabulary size**
  - Group **similar words** together
  - Improve **classification accuracy**
- **Examples**
- eating → eat
- ate → eat
- adjustable → adjust
- talking → talk



Stemming

- Uses **fixed rules** to remove suffixes/prefixes
- **No knowledge** of the language
- Simple, fast, but may produce meaningless words
- **Examples:**
- talking → talk
- adjustable → adjust
- ability → abil ❌ (wrong)
- **Definition:** *Stemming = Basic chopping of affixes using rules.*

Reference Models / Algorithms

- **Porter Stemmer (1980)** → oldest, widely used, suffix stripping only.
- **Snowball Stemmer (Porter2, 2001)** → improved version, suffix focus.
- **Lancaster Stemmer (Paice/Husk, 1990)** → more aggressive, strips prefixes too.
- **Lovins Stemmer (1968)** → one of the first, long list of suffix rules.
- **Key Takeaway**
- **Suffix-only stemmers** (Porter, Snowball) = conservative, safer.
- **Prefix + suffix stemmers** (Lancaster) = aggressive, risk of over-stemming.
- Choice depends on **task requirements** (IR/search vs linguistics-heavy NLP).

Lemmatization

- Uses **linguistic knowledge** (dictionary + grammar rules)
- Produces real base words (**lemmas**)
- More accurate, but computationally heavier
- **Examples:**
- ate → eat
- better → well
- ability → ability ✅
- **Definition:** *Lemmatization = reducing words to lemma using language knowledge.*

What is POS?

- **Part of Speech (POS):** Category of words based on their role in a sentence.
- Common POS categories (English grammar):
  - **Noun** → person, place, thing (*Elon, Mars, fruits*)
  - **Verb** → action (*eat, play, fly*)
  - **Pronoun** → replaces noun (*he, she, they*)
  - **Adjective** → describes noun (*red car, sweet fruit*)
  - **Adverb** → describes verb/adjective (*quickly ran, always studies*)
  - **Conjunction** → joins phrases (*and, but, or*)
  - **Preposition** → links noun with another (*in, on, at*)

Why POS Matters in NLP?

- Helps in:
  - **Information Extraction** (nouns, verbs)
  - **Sentiment Analysis** (adjectives, adverbs)
  - **Entity Recognition** (proper nouns)
  - **Grammar Checking**
  - **Summarization & Translation**

Code Demo