Lecture 5 - Introduction to NLP and Text Processing

Structured Summary

Natural Language Processing

- Natural Language Processing (NLP) combines computational techniques with linguistic theory to analyze and process human language.
- **Daily Use Cases**: Search engines (e.g., Google queries about U.S. presidents), transcription tools (e.g., Samsung Galaxy Note's handwriting-to-text).
- **Typical Tasks**:
    - Question answering (e.g., "Who is the 48th U.S. president?").
    - Entity recognition (e.g., identifying names like "Donald Trump").
    - Text normalization and translation.

Definitions and Concepts

- **Natural vs. Artificial Languages**: Natural languages evolve organically (e.g., English), while artificial languages follow strict rules (e.g., programming languages).
- **Semantics**: Meaning derived from words and sentences.
- **Pragmatics**: Contextual interpretation of language.

> **Definition of NLP, Liddy, E.D. (2001)**
>
> "Natural Language Processing is a theoretically motivated range of **computational techniques** for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of **achieving human-like language processing** for a **range of tasks or applications**."

Challenges for NLP:

- Paraphrasing, translation, question answering, and inference require understanding context and ambiguity.
- Example: Translating idioms (e.g., "break a leg") → cultural nuances.

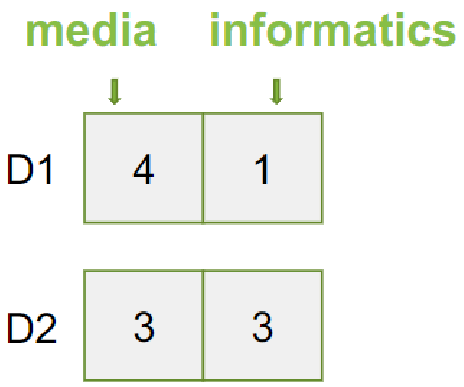
Algorithms and Techniques

- **High-Level Tasks**: Sentiment analysis, summarization, topic modeling.
- **Low-Level Tasks**: Tokenization, stemming, stop word removal.

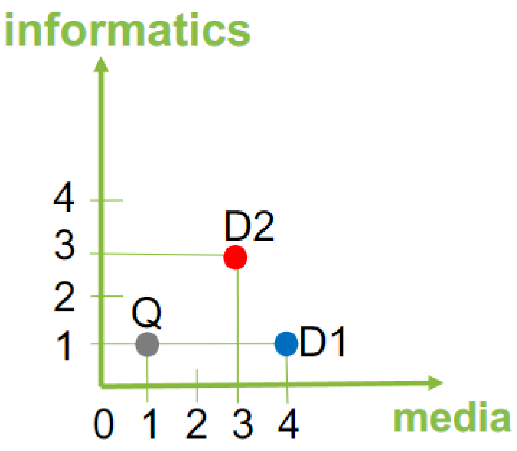
Python Example: Counting word occurrences

Converting text documents into word frequency vectors for search relevance.

- **Relevance Calculation:** By turning words into numerical features (counts), search algorithms can compare documents (e.g.: the distance between D1 and D2) to the query in a structured way.
- **Vector Space Model Foundation:** This is the basis of many Information Retrieval systems and a stepping stone toward more advanced **Natural Language Processing** techniques



	media	informatics
D1	4	1
D2	3	3



Tokenization and Normalization

- **Tokenization**: Splitting text into words/tokens, handling punctuation and filler or stop word like: the, and, a, to ...
  - **Morphological Variants:** Words like “run,” “runs,” and “running” may refer to the same concept but won’t match exactly.
  - **Synonyms:** Different words with the same meaning (“car” vs. “automobile”) will be overlooked.
  - **Context & Semantics:** Token-level matching doesn’t account for context or how words interact to convey meaning.
  - Upper case / lower case letters (e.g., all lower case)
    - Tricky if upper case is required to detect names, grammar
  - Acronyms (U.K. -> UK)
  - Expanding contractions ("don't" -> "do not”).
  - Umlauts (für -> fuer or fuer -> für)
  - Dealing with numbers and symbols in text (“three” -> “3”)
  - Correcting misspelling
- **Text Normalization**: Lowercasing, expanding contractions (e.g., "don’t" → "do not"), handling umlauts (e.g., "fütterung" → "fuetterung") and the other problems mentioned.

> **Handling Tokenization Challenges**
>
> Tokenizing phrases like "Dr. Mayer-Hauser" requires handling abbreviations and compound words. Advanced tokenizers (e.g., spaCy’s) use rules to split such cases accurately.

> **Stop Word Removal Nuances**
>
> While removing stop words improves efficiency, it can lose context (e.g., "to be or not to be" becomes "be not be"). Domain-specific stop word lists may be needed.

Stemming vs. Lemmatization

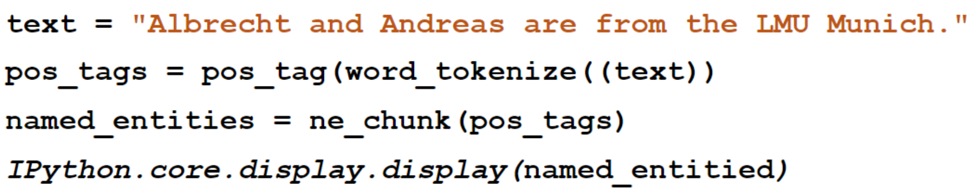
- **Stemming**: Heuristically reducing words to roots (e.g., "running" → "run"). Uses algorithms like Porter Stemmer.
  - Applies crude heuristic rules to strip suffixes (e.g., “running” → “run”)
  - Does not guarantee valid words in the language
  - Faster but can be less accurate

- **Lemmatization**: Linguistically accurate base forms (e.g., "was" → "be"). Relies on dictionaries and grammar rules.
    - Uses vocabulary and morphological analysis (e.g., “running” → “run” if it’s a verb)
        - Produces valid dictionary forms (“run” as a base form)
        - Slower but more linguistically accurate

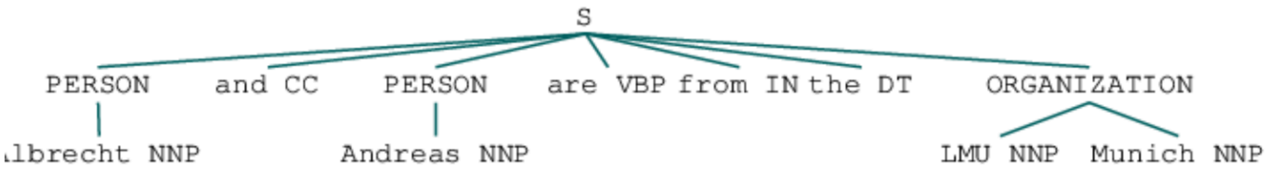
Implementing Stemming/Lemmatization with ML

Machine Learning can improve stemming/lemmatization by training models on annotated corpora to predict morphological roots, reducing reliance on rule-based systems.

Named Entity Recognition (NER)



```
text = "Albrecht and Andreas are from the LMU Munich."
pos_tags = pos_tag(word_tokenize((text))
named_entities = ne_chunk(pos_tags)
IPython.core.display.display(named_entitied)
```



- Identifies entities (e.g., people, organizations) in text.
- Example: "LMU Munich" → tagged as `ORGANIZATION`.
- Tools: spaCy, Stanford NER.

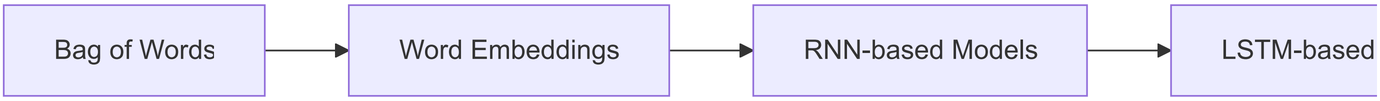
Corpus and Corpora

- **Corpus**: A structured collection of texts (e.g., Project Gutenberg).
- **Parallel Corpora**: Texts in multiple languages (e.g., translated books).

Bag of Words:

The Bag of Words (BoW) model represents text as word frequencies, ignoring grammar and order. While simple, it underpins tasks like spam detection. For example:

- "The cat sleeps. The dog barks." → {"the":2, "cat":1, "sleeps":1, "dog":1, "barks":1}. Limitations include missing semantic relationships (e.g., "not good" vs. "good").


Bag of Words
Word Embeddings
RNN-based Models
LSTM-based