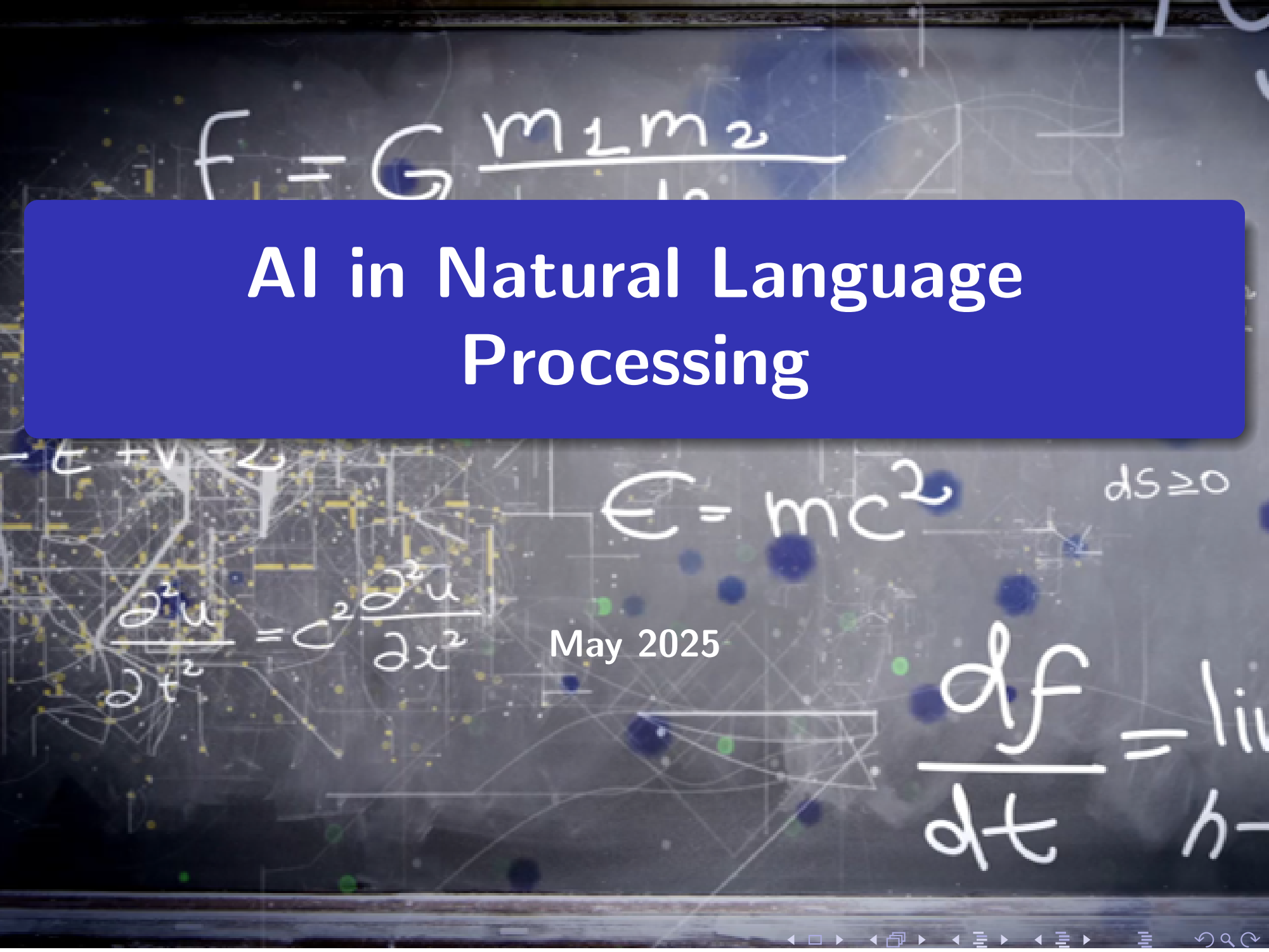


AI in Natural Language Processing

May 2025

Topics to be discussed

Part I. Introduction

1. Introduction to Artificial intelligence

Part II. Search and optimisation.

2. Search - basic approaches
3. Search - optimisation
4. Two-player deterministic games
5. Evolutionary and genetic algorithms

Part III. Machine learning and data analysis.

6. Regression, classification and clustering (Part I & II)
8. Artificial neural networks
9. Bayesian models
10. Reinforcement Learning

Part IV. Logic, Inference, Knowledge Representation

11. Propositional logic and predicate logic
12. Knowledge Representation

Part V. AI in Action: Language, Vision

13. AI in Natural language processing
14. AI in Vision and Perception

Part VI. Summary

15. AI engineering, Explainable AI, Ethics

Natural language processing and AI

Natural Language Processing (NLP) is a subfield of artificial intelligence (AI) that focuses on the interaction between computers and human language. It involves the development of algorithms and models to enable computers to understand, interpret, and generate natural language.

Components of NLP:

- **Text Processing:** Preprocessing and cleaning of text data, including tasks such as tokenization, stemming, and normalization.
- **Syntax and Grammar:** Parsing and analyzing the grammatical structure of sentences to understand the relationships between words.
- **Semantics and Meaning:** Extracting the meaning of text, including word sense disambiguation and semantic role labeling.
- **Discourse and Pragmatics:** Understanding the context, coherence, and intentions behind a piece of text.

This lecture is focused mostly on Text Processing.

Knowledge Representation in NLP:

- **Lexical Resources:** Creating and maintaining lexical databases, such as WordNet, that store information about words, their meanings, and relationships.
- **Semantic Networks:** Representing knowledge using networks of concepts and their relationships, allowing for semantic reasoning and inference.
- **Ontologies:** Formal representations of knowledge in specific domains, capturing concepts, relationships, and constraints.
- **Statistical Models:** Utilizing statistical methods and machine learning techniques to learn patterns and extract knowledge from large-scale text corpora.

Challenges in Knowledge Representation:

- **Ambiguity:** Resolving ambiguity in natural language, such as word sense ambiguity and referential ambiguity.
- **Scalability:** Handling large-scale knowledge representation and reasoning in complex domains.
- **Subjectivity:** Incorporating subjective and contextual aspects of language understanding.

Applications of NLP:

- **Machine Translation:** Automatically translating text from one language to another.
- **Information Extraction:** Extracting structured information from unstructured text, such as named entity recognition and relation extraction.
- **Text Summarization:** Generating concise summaries of long documents or articles.
- **Sentiment Analysis:** Analyzing and determining the sentiment expressed in text, such as positive, negative, or neutral sentiment.
- **Question Answering:** Answering questions posed in natural language, often by extracting information from relevant sources.

Emerging Areas in NLP:

- **Conversational AI:** Developing intelligent chatbots and virtual assistants capable of engaging in natural language conversations.
- **Ethical and Bias Considerations:** Addressing issues of bias, fairness, and ethical concerns in language models and NLP applications.
- **Multilingual and Cross-lingual NLP:** Extending NLP techniques to support multiple languages and enabling cross-lingual understanding and analysis.

Natural Language Processing (NLP)

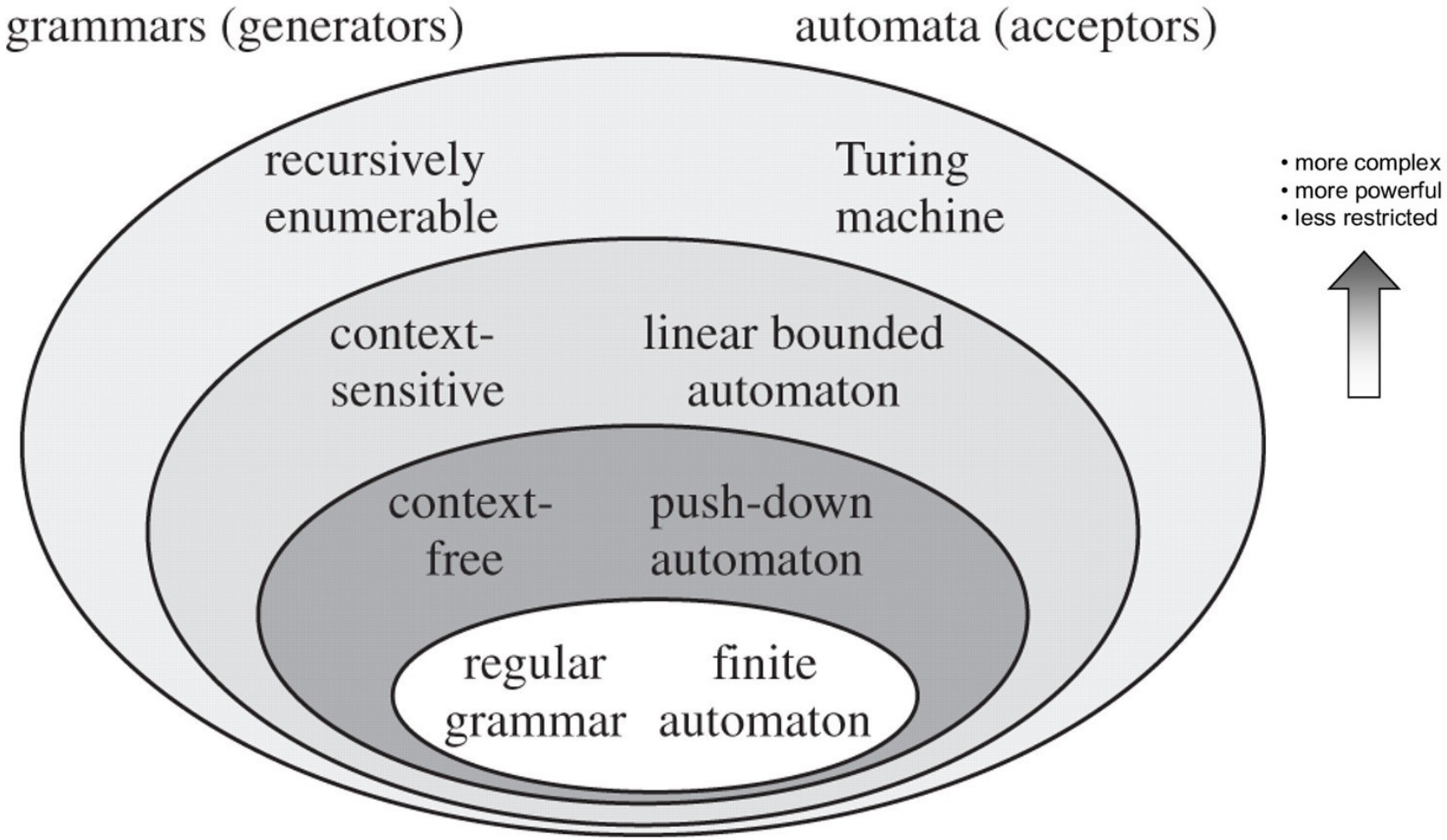
Natural Language Understaning (NLU) and Natural Language Generation (NLG)

In the field of Artificial Intelligence (AI), Natural Language Processing (NLP) focuses on teaching machines to understand and interpret human language. It encompasses two main parts:

- **Natural Language Understanding (NLU)**
  - NLU is concerned with the semantic analysis or understanding of the intended meaning of the message or text.
  - It involves techniques such as language modeling, sentiment analysis, named entity recognition, and text classification.
  - NLU aims to enable machines to comprehend and extract useful information from human language.
- **Natural Language Generation (NLG)**
  - NLG focuses on the generation of a message or text by a machine.
  - It involves techniques such as text summarization, machine translation, dialogue generation, and content generation.
  - NLG aims to enable machines to produce human-like language that is coherent and contextually appropriate.

These parts of NLP play a crucial role in various applications. For example, conversational agents like Apple Siri utilize NLP to listen to user queries and find relevant answers. However, recent years have witnessed the emergence of more sophisticated models in the market, such as Gemini, GPT-4, and other advanced language models. These models push the boundaries of NLP by leveraging large-scale pre-training and fine-tuning techniques, resulting in impressive language generation capabilities.

NLP - (classical) Chomsky hierarchy reminder



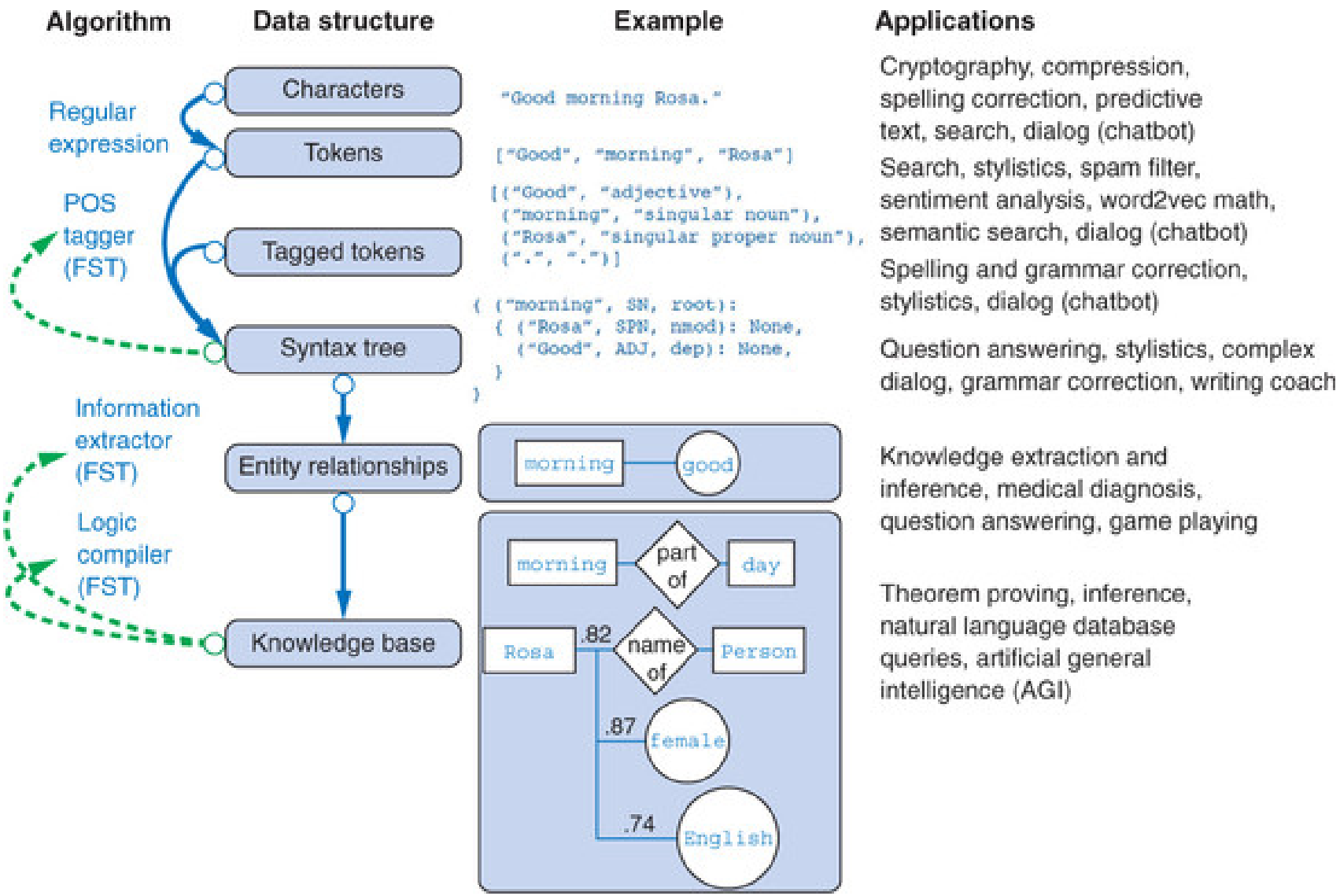
https://devopedia.org/chomsky-hierarchy

Key AI Techniques in NLP

- **1. Rule-based Methods:** These techniques rely on predefined linguistic rules and patterns to process and analyze natural language data.
- **2. Statistical Methods:** Statistical approaches utilize probabilistic models and algorithms to extract meaning from text. These methods often involve techniques such as Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs).
- **3. Machine Learning (ML) Methods:** ML algorithms can be employed to automatically learn patterns and rules from labeled or unlabeled data. Supervised learning algorithms like Naive Bayes, Support Vector Machines (SVM), and Neural Networks are commonly used in NLP.
- **4. Deep Learning Methods:** Deep Learning techniques, particularly Neural Networks with multiple layers, have gained significant attention in NLP. Models like Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), and Transformer-based architectures (e.g., BERT) have shown remarkable performance in various NLP tasks.
- **5. Probabilistic Graphical Models (PGMs):** PGMs, including Bayesian Networks and Markov Random Fields, provide a framework for modeling complex dependencies in natural language data. They enable effective representation and inference in NLP applications.
- **6. Reinforcement Learning (RL):** RL techniques involve an agent learning to make decisions in an environment through trial and error. RL can be applied in NLP for tasks like dialogue systems and text generation.
- **7. Hybrid Approaches:** Combining multiple AI techniques, such as integrating rule-based methods with deep learning models, can leverage the strengths of each approach and achieve improved performance in NLP.

Natural Language Processing

An example of NLP pipeline (symbolic)



Source: The book [2]

Preprocessing techniques in Natural Language Processing (NLP) refer to the initial steps of cleaning and transforming unstructured text data into a format that can be easily analyzed. These techniques include tokenization, stemming, and stop-word removal.

Consider the sentence:

"I am planning to buy a new laptop".

The preprocessing steps involved in analyzing this sentence could include **tokenization** *(breaking the sentence into individual words),* **stemming** *(reducing words to their root form), and* **stop-word removal** *(eliminating common words like "am" and "to").*

For example, consider the sentence:

"The cat in the hat".

After tokenization, the sentence would be represented as a list of tokens:

["The", "cat", "in", "the", "hat"].

Text Preprocessing and Text Representation

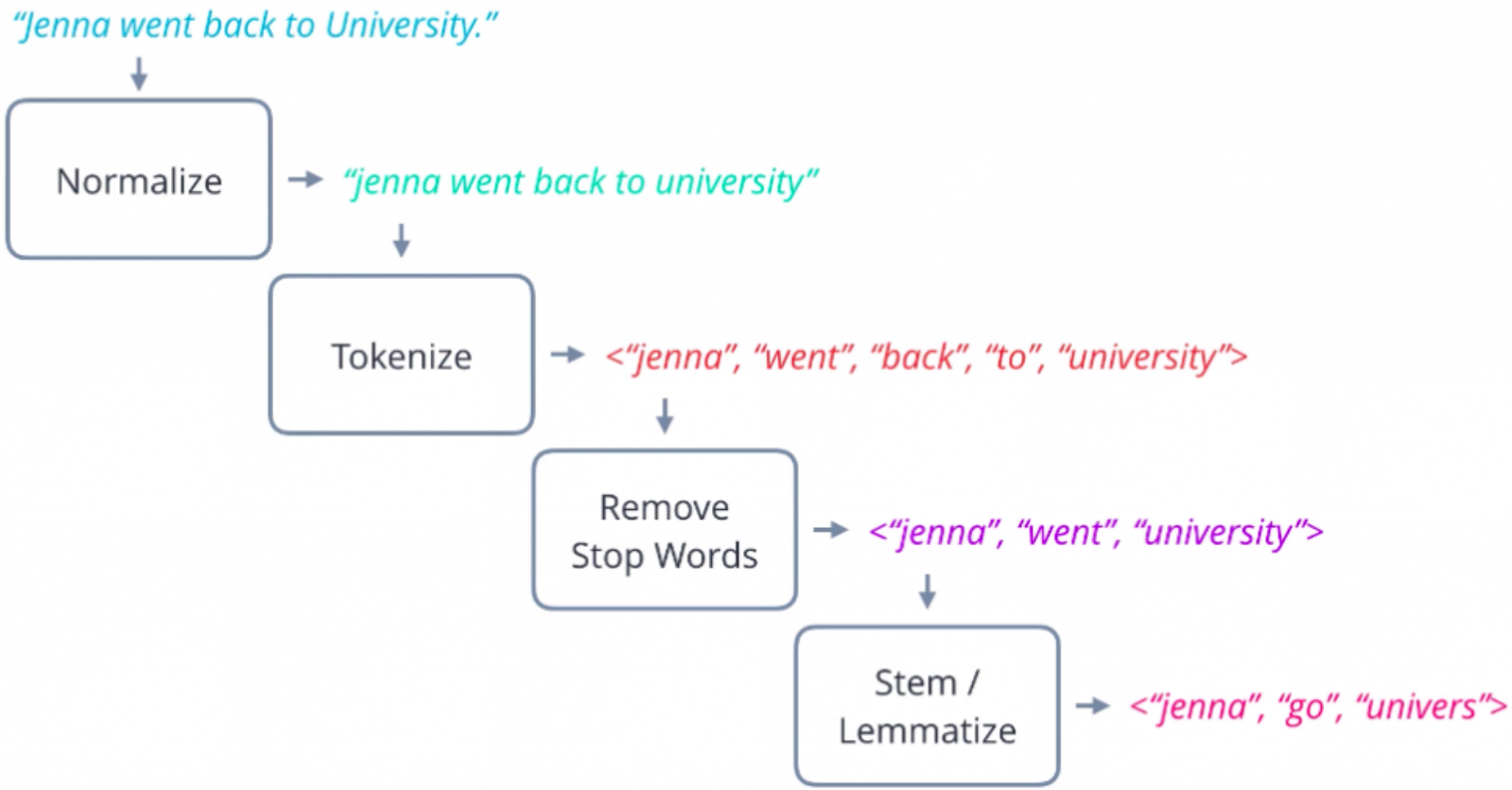
Preprocessing Techniques (cont.)

In Natural Language Processing (NLP), preprocessing and text representation are crucial steps in preparing textual data for analysis and modeling.

1. Tokenization: Breaking down text into individual units called tokens.
2. Text Normalization: Transforming text into a standardized format by removing punctuation, converting to lowercase, etc.
3. Stop Word Removal: Eliminating commonly occurring words that do not contribute much to the overall meaning of the text.
4. Stemming: Reducing words to their root or base form by removing prefixes or suffixes.
5. Lemmatization: Obtaining the base form of a word using vocabulary and morphological analysis.
6. Feature Extraction: Transforming text into a numerical representation, such as Bag-of-Words, TF-IDF, or n-gram models.
7. Word Embeddings: Dense vector representations of words capturing semantic relationships.

Text Preprocessing and Text Representation

From normalization to stemming - An example



https://medium.com/predict/how-does-nlp-pre-processing-actually-work-8d097c179af1

Text Preprocessing and Text Representation

Tokenization and Normalization

Tokenization

- Tokenization is the process of breaking down a text into individual units called tokens.
- These tokens can be words, sentences, or even subword units, depending on the granularity required for the task at hand.

Example:

- Input Text: "I love to eat ice cream!"
- Tokenized Output: ["I", "love", "to", "eat", "ice", "cream", "!"]

Text Normalization

Text normalization aims to transform text into a standardized format:

- Converting text to lowercase to ensure case-insensitive processing.
- Removing punctuation marks, special characters, and symbols that do not contribute much to the overall meaning.
- Handling numbers and dates by converting them to a common format.
- Resolving contractions (e.g., "can't" to "cannot") for consistent representation.

Text normalization ensures consistency and reduces the dimensionality of the data, enabling effective analysis and modeling.

Example:

- Input Text: "I'm happy today! Let's celebrate."
- Normalized Output: "i am happy today lets celebrate"

Text Preprocessing and Text Representation

Types of Tokens

Tokenization is the process of breaking down a text into smaller units, called tokens. While we commonly associate tokens with words, tokenization can be applied to various units of text. Let's explore the different types of tokens:

- **Characters** The most basic unit of tokenization is characters. In this approach, each individual character in the text is considered a token. Character-level tokenization is useful in tasks like text generation or language modeling, where the model needs to learn from individual characters.
- **Words**
  Words are one of the most commonly used units of tokenization. Each word in the text is considered a token. Word-level tokenization is beneficial for tasks such as text classification, information retrieval, and sentiment analysis, where the meaning and context of words are important.
- **Sentences**
  Tokenizing at the sentence level involves breaking the text into individual sentences. Each sentence becomes a token. Sentence-level tokenization is useful in tasks like text summarization, machine translation, and sentiment analysis, where the analysis needs to be done on a per-sentence basis.
- **Lines**
  Tokenizing at the line level involves considering each line of the text as a token. Line-level tokenization is common in tasks like document processing or poetry analysis, where the structure and formatting of the text play a role.
- **Paragraphs**
  Tokenizing at the paragraph level involves treating each paragraph as a token. Paragraph-level tokenization is useful in tasks like document classification or text generation, where the content and structure of paragraphs are significant.
- **N-grams**
  N-grams are contiguous sequences of N items, such as words or characters. Tokenizing at the N-gram level involves extracting these N-gram sequences from the text. N-gram tokenization is beneficial in tasks like language modeling or information retrieval, where the sequence of words or characters is important.

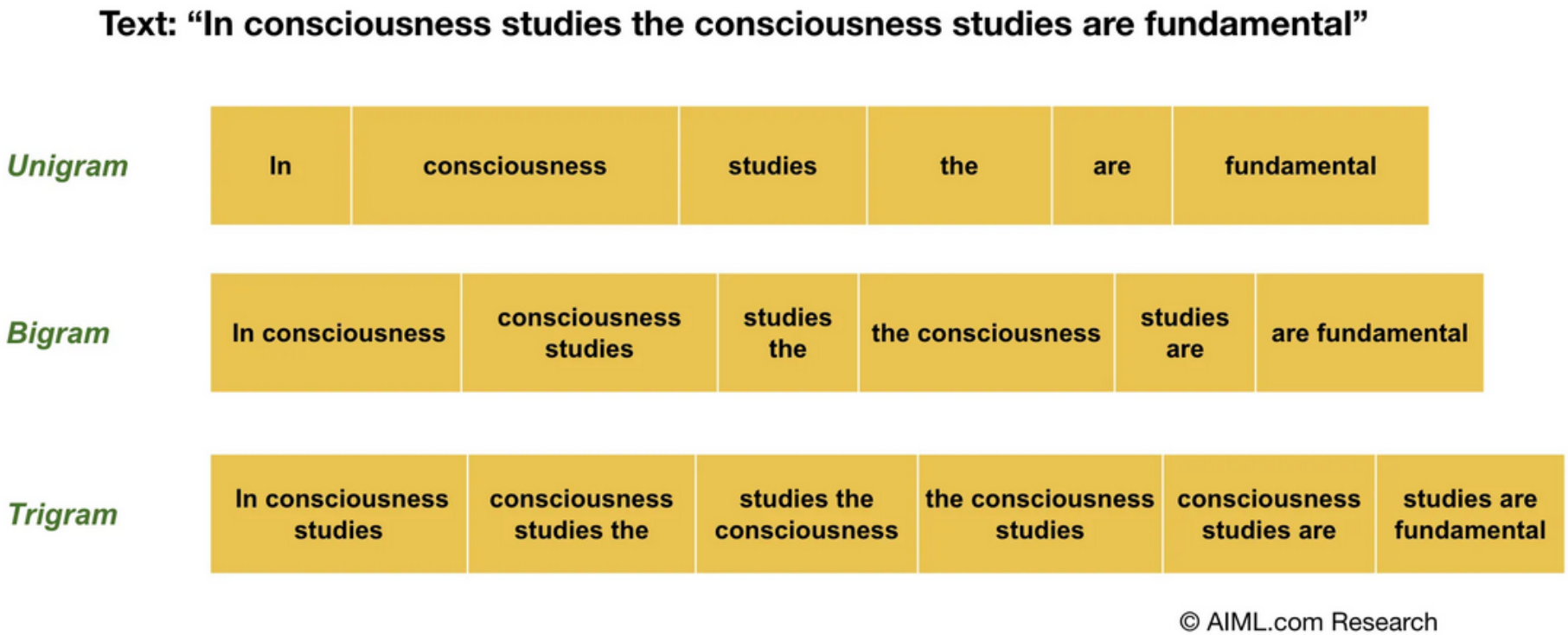
By considering different types of tokens, we can apply tokenization at various levels of granularity to suit the specific requirements of our NLP tasks.

- **N-gram Model:**
  The n-gram model represents sequences of consecutive words of length n. It considers the local context of words and captures the co-occurrence patterns within a fixed window.
- **For example:**
  - In a trigram model (n=3), the sentence "I love natural language processing" would be represented as "I love natural", "love natural language", "natural language processing".

  N-gram models are simple and efficient, but they may not capture long-range dependencies and suffer from data sparsity issues when n is large.
- **Skip-gram Model:**
  The skip-gram model, popularized by Word2Vec, focuses on learning distributed representations of words by predicting context words given a target word. It aims to capture the semantic relationships between words by considering the surrounding context.
- **For example:**
  - Given the sentence "I love natural language processing," the skip-gram model would predict "love" and "processing" given the target word "natural".

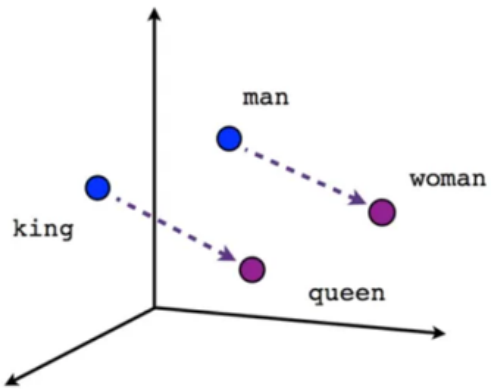
  Skip-gram models are effective at capturing word semantics and can handle large vocabularies. However, they can be computationally expensive.

N-gram model

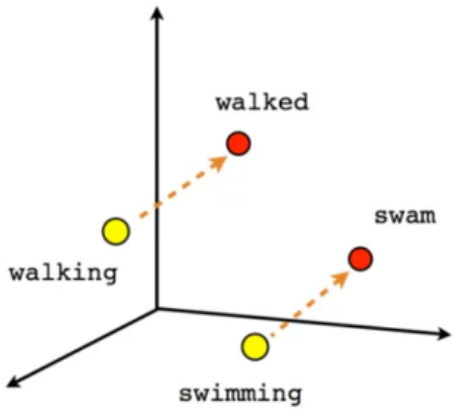


https://aiml.com/what-is-an-n-gram-model/

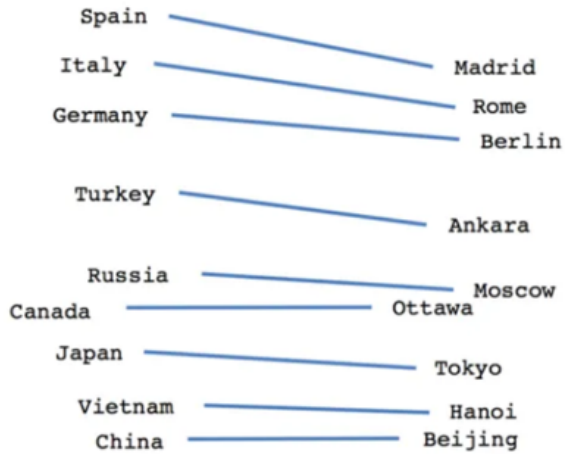
Word2vec model



Male-Female



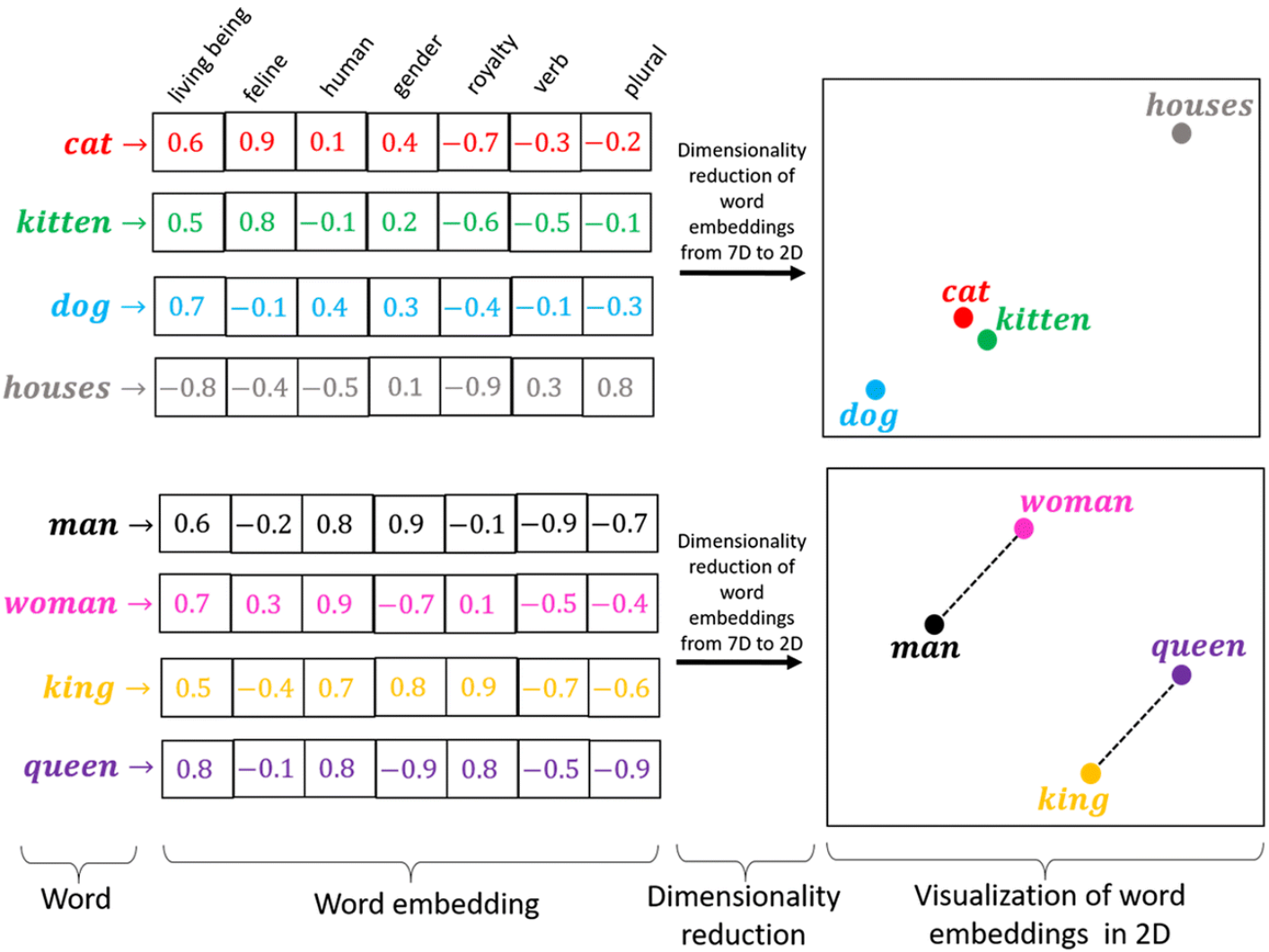
Verb tense



Country-Capital

https://aiml.com/what-is-an-n-gram-model/

Word2vec model



https://aiml.com/what-is-an-n-gram-model/

Text Preprocessing and Text Representation

Regular Expressions in NLP and Tokenization

Regular Expressions are powerful tools for pattern matching and string manipulation.

- In the context of natural language processing (NLP), regular expressions are commonly used for tasks such as tokenization.
- Regular expressions allow us to define patterns to identify and split text into meaningful tokens.

Tokenization

- Tokenization is the process of breaking down text into smaller meaningful units, called tokens.
- Tokens can be words, sentences, or any other defined units depending on the task.
- Regular expressions play a crucial role in tokenization by providing patterns to match and split the text into tokens.
- For example, we can use regular expressions to identify word boundaries or punctuation marks as tokenization delimiters.

Example: Tokenization using Regular Expressions

- Consider the sentence: "I love NLP! It's fascinating."
- Using a regular expression pattern like `[\w']+`, we can tokenize the sentence into the following tokens: ["I", "love", "NLP", "It's", "fascinating"].
- The regular expression pattern `[\w']+` matches one or more word characters or apostrophes, effectively splitting the sentence into individual words.

Regular expressions

Regular Expressions (Regex) Cheat Sheet

Special Characters in Regular Expressions & their meanings

Character	Meaning	Example
*	Match **zero, one or more** of the previous	`Ah*` matches "Ahhhhh" or "A"
?	Match **zero or one** of the previous	`Ah?` matches "Al" or "Ah"
+	Match **one or more** of the previous	`Ah+` matches "Ah" or "Ahhh" but not "A"
\	Used to **escape** a special character	`Hungry\?` matches "Hungry?"
.	Wildcard character, matches **any** character	`do.*` matches "dog", "door", "dot", etc.
( )	**Group** characters	See example for |
[ ]	Matches a **range** of characters	`[cbf]ar` matches "car", "bar", or "far" `[0-9]+` matches any positive integer `[a-zA-Z]` matches ascii letters a-z (uppercase and lower case) `[^0-9]` matches any character not 0-9.
|	Matche previous **OR** next character/group	`(Mon)|(Tues)day` matches "Monday" or "Tuesday"
{ }	Matches a specified **number of occurrences** of the previous	`[0-9]{3}` matches "315" but not "31" `[0-9]{2,4}` matches "12", "123", and "1234" `[0-9]{2,}` matches "1234567..."
^	**Beginning** of a string. Or within a character range `[]` negation.	`^http` matches strings that begin with http, such as a url. `[^0-9]` matches any character not 0-9.
$	**End** of a string.	`ing$` matches "exciting" but not "ingenious"

See also: **Regular Expression Character Classes CheatSheet**.

This is a work in progress - Questions, comments, criticism, or requests can be directed Here

Text Preprocessing and Text Representation

POS Generation from N-grams

1. **N-gram Models**:
   - N-gram models are statistical models that capture the probability distribution of sequences of words in a text.
   - An n-gram refers to a contiguous sequence of n items, which can be words, characters, or other linguistic units.
2. **Training the N-gram Model**:
   - To train the model, annotated text with word-POS tag pairs is used.
   - The training process involves collecting n-grams of different sizes and counting their occurrences in the annotated corpus.
   - The frequency of each n-gram is calculated, which helps estimate the probability of encountering a particular n-gram in the text.
3. **Generating POS Tags**:
   - Once the n-gram model is trained, it can be used to generate POS tags for unseen text.
   - The unseen text is broken down into n-grams, and the model predicts the most likely POS tags for each n-gram based on the learned probabilities.
   - The generated POS tags provide information about the grammatical category and syntactic role of each word in the text.

Generating accurate POS tags can be challenging due to the ambiguity of natural language and the presence of out-of-vocabulary words. Smoothing techniques, handling unknown words through techniques like morphological analysis or character-level models, and incorporating linguistic rules can improve accuracy. Advanced models like Hidden Markov Models (HMMs) or Conditional Random Fields (CRFs) can enhance performance by considering the contextual dependencies among POS tags.

Text Preprocessing and Text Representation

Grammar Tree Generation from N-grams and POS

1. **Grammar Tree**:
   - A grammar tree represents the syntactic structure of a sentence, showing how words and phrases are related.
   - It can be generated by combining the POS tags and the corresponding words in a sentence.
   - Each node in the tree represents a word or phrase, and the edges indicate the relationships between them.
2. **Generation Process**:
   - Using the n-gram model, the most likely POS tags for each word in a sentence can be predicted.
   - The POS tags and words are combined to form the initial grammar tree.
   - Additional syntactic rules and constraints can be applied to refine and structure the grammar tree further.
3. **Applications and Challenges**:
   - Grammar tree generation is essential for tasks like syntactic parsing, text-to-speech synthesis, and grammar checking.
   - Challenges include handling ambiguity, resolving parsing ambiguities, and capturing complex sentence structures accurately.
   - Advanced techniques like probabilistic context-free grammars (PCFGs) and dependency parsing algorithms can improve the accuracy of grammar tree generation.

Stop word removal and stemming are preprocessing steps to improve the efficiency and effectiveness of text analysis. By removing irrelevant words and reducing words to their base forms, these techniques improve the quality of textual data for further analysis, such as information retrieval, text classification, and topic modeling.

Stop Word Removal
Stop words are commonly occurring words in a language that do not carry significant meaning in the context of a particular text analysis task. These words, such as "the," "is," and "and," are often removed to reduce noise and focus on more relevant terms.

Example:

- Input Text: "I want to go to the park and play with my dog."
- Stop Word Removed Output: "want go park play dog"

Stemming
Stemming is the process of reducing words to their root or base form. It involves removing prefixes or suffixes from words to obtain their core meaning. Stemming helps in handling variations of words and reducing the vocabulary size.

Example:

- Input Word: "running"
- Stemmed Output: "run"

Text Preprocessing and Text Representation

Feature Extraction

Feature extraction techniques enable numerical representation of text data, allowing machine learning algorithms to work with textual information.

Bag-of-Words (BoW)
The Bag-of-Words model represents a document as a collection of words without considering the word order. It creates a vocabulary of unique words in the corpus and represents each document as a vector indicating the presence or absence of these words.

Example:

- Corpus: ["I love dogs", "I hate cats", "I love cats"]
- Vocabulary: ["I", "love", "dogs", "hate", "cats"]
- Document Vectors:
  - "I love dogs": [1, 1, 1, 0, 0]
  - "I hate cats": [1, 0, 0, 1, 1]
  - "I love cats": [1, 1, 0, 0, 1]

Term Frequency-Inverse Document Frequency (TF-IDF)
TF-IDF measures the importance of a word in a document by considering both its frequency in the document (Term Frequency) and its rarity in the entire corpus (Inverse Document Frequency). It assigns higher weights to words that are more informative.

n-gram Models
n-gram models capture the relationship between adjacent words in a document. An n-gram is a contiguous sequence of n words. Common choices include unigrams (individual words), bigrams (pairs of adjacent words), and trigrams (triplets of adjacent words).

Text Preprocessing and Text Representation

Term Frequency - Inverse Document Frequency (TF-IDF)

TF-IDF is a popular weighting scheme used to measure the importance of words in a document within a corpus. It combines the concepts of Term Frequency (TF) and Inverse Document Frequency (IDF) to create document vectors. Let's understand how TF and IDF are calculated:

- **Term Frequency (TF)**
  TF measures the frequency of a word in a particular document. It is calculated using the formula:

$$\text{TF} = \frac{\text{No. of repeated words in sentence}}{\text{No. of words in sentence}}$$

  The TF value indicates how often a word appears in a document relative to the total number of words in that document. A higher TF value signifies a greater relevance of the word within the document.

- **Inverse Document Frequency (IDF)**
  IDF measures the rarity or importance of a word across the entire corpus. It is calculated using the formula:

$$\text{IDF} = \log\left(\frac{\text{No. of sentences}}{\text{No. of sentences containing word}}\right)$$

  The IDF value quantifies the informativeness of a word by considering its presence in a subset of sentences compared to its presence in the entire corpus. Words that occur in fewer sentences across the corpus receive higher IDF scores.

- **TF-IDF Score (w)**
  The TF-IDF score for a word is computed by multiplying its TF value with its IDF value. Mathematically, it can be expressed as:

$$w = \text{TF} \times \text{IDF}$$

  The TF-IDF score allows us to assign a weight to each word in a document based on its frequency in the document (TF) and its rarity across the corpus (IDF). Words that occur frequently in a document but rarely in the corpus will have higher TF-IDF scores, indicating their significance to that specific document.

Text Preprocessing and Text Representation

TF-IDF Calculation - An Example

To better understand TF-IDF, let's consider an example with a document corpus consisting of three documents:

1. Document 1: "The cat sat on the mat"
2. Document 2: "The dog chased the cat"
3. Document 3: "The mouse ran across the floor"

We will calculate the TF-IDF scores for the word "cat" in each document.

- **Term Frequency (TF)**
  The term frequency for "cat" in each document is as follows:
  Document 1: TF("cat") $= 1/6$
  Document 2: TF("cat") $= 1/5$
  Document 3: TF("cat") $= 0/6$ (no occurrence of "cat")
- **Inverse Document Frequency (IDF)**
  The inverse document frequency for "cat" is calculated as:
  IDF("cat") $= \log\left(\frac{3}{2}\right) = 0.405$ (approximately)
- **TF-IDF Score**
  Finally, multiplying the TF and IDF values, we obtain the TF-IDF scores for "cat":
  Document 1: TF-IDF("cat") $= (1/6) \times 0.405$
  Document 2: TF-IDF("cat") $= (1/5) \times 0.405$
  Document 3: TF-IDF("cat") $= 0$ (since "cat" does not occur in Document 3)

The TF-IDF scores provide a measure of the importance of the word "cat" in each document based on its frequency within the document and rarity across the corpus.

Word embeddings are real-valued vector representations of words in a high-dimensional space. They aim to capture semantic and syntactic relationships between words based on their contextual usage. Formally, given a vocabulary V of size $|V|$, a word embedding is a function $E : V \to \mathbb{R}^d$, where d represents the dimensionality of the word vectors.

Example: Word2Vec

One popular algorithm for generating word embeddings is Word2Vec. It utilizes a shallow, two-layer neural network that takes a large corpus of text as input and learns word vectors. It optimizes the objective function based on the likelihood of predicting words in the context of other words. Mathematically, Word2Vec maximizes the following objective:

$$\sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log P(w_{t+j} | w_t),$$

where T is the total number of words in the corpus, c is the window size, and $P(w_{t+j}|w_t)$ represents the probability of word w_{t+j} occurring in the context of word w_t.

Example: GloVe

Another commonly used word embedding technique is GloVe (Global Vectors for Word Representation). It leverages a co-occurrence matrix to capture the statistical properties of words and their contexts. GloVe learns word vectors by minimizing the following objective:

$$\sum_{i=1}^{|V|} \sum_{j=1}^{|V|} f(X_{ij}) \left(E_i \cdot E_j + b_i + b_j - \log X_{ij} \right)^2,$$

where E_i and E_j are the word vectors, b_i and b_j are the biases, X_{ij} denotes the co-occurrence count between words i and j, and f is a weighting function.

Text Preprocessing and Text Representation

Vector Similarity in NLP

In natural language processing (NLP), vector similarity plays a crucial role in various tasks. It involves measuring the similarity or closeness between two vectors representing textual data. Let's explore some methods, applications, and examples of vector similarity in NLP.

- **Cosine Similarity:** Cosine similarity measures the cosine of the angle between two vectors. It is widely used to determine the similarity between documents, sentences, or words in NLP. Applications include document similarity analysis, information retrieval, and search engine ranking.
- **Euclidean Distance:** Euclidean distance calculates the straight-line distance between two vectors in a Euclidean space. It can be utilized to measure the dissimilarity or similarity between textual data points. It finds applications in clustering, anomaly detection, and nearest neighbor search.
- **Jaccard Similarity:** Jaccard similarity measures the size of the intersection divided by the size of the union of two sets. In NLP, it is commonly used for comparing sets of words or terms, such as in document clustering, keyword extraction, and text summarization.
- **Word Embedding Similarity:** Word embeddings represent words as dense vectors in a continuous space, capturing semantic relationships. Similarity between word vectors can be calculated using cosine similarity or Euclidean distance. Word2Vec, GloVe, and FastText are popular word embedding models.
- **Document Embedding Similarity:** Document embeddings capture the semantic meaning of entire documents. Similarity between document vectors can be computed using cosine similarity or Euclidean distance. It enables tasks like document clustering, information retrieval, and recommendation systems.

These methods of vector similarity in NLP enable us to quantify the similarity or dissimilarity between textual data points, allowing us to perform a wide range of NLP tasks more effectively.

Text Preprocessing and Text Representation

Vector Similarity Measures - Math Formulas

In natural language processing (NLP), various vector similarity measures are used to quantify the similarity between textual data. Let's examine the math formulas for some of these measures:

- **Cosine Similarity:**
  The cosine similarity between two vectors **A** and **B** is calculated using the dot product and vector norms:

$$\text{Cosine Similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

- **Euclidean Distance:**
  The Euclidean distance between two vectors **A** and **B** is computed as:

$$\text{Euclidean Distance}(\mathbf{A}, \mathbf{B}) = \sqrt{\sum_{i=1}^{n} (A_i - B_i)^2}$$

- **Jaccard Similarity:**
  The Jaccard similarity between two sets A and B is given by the ratio of their intersection to their union:

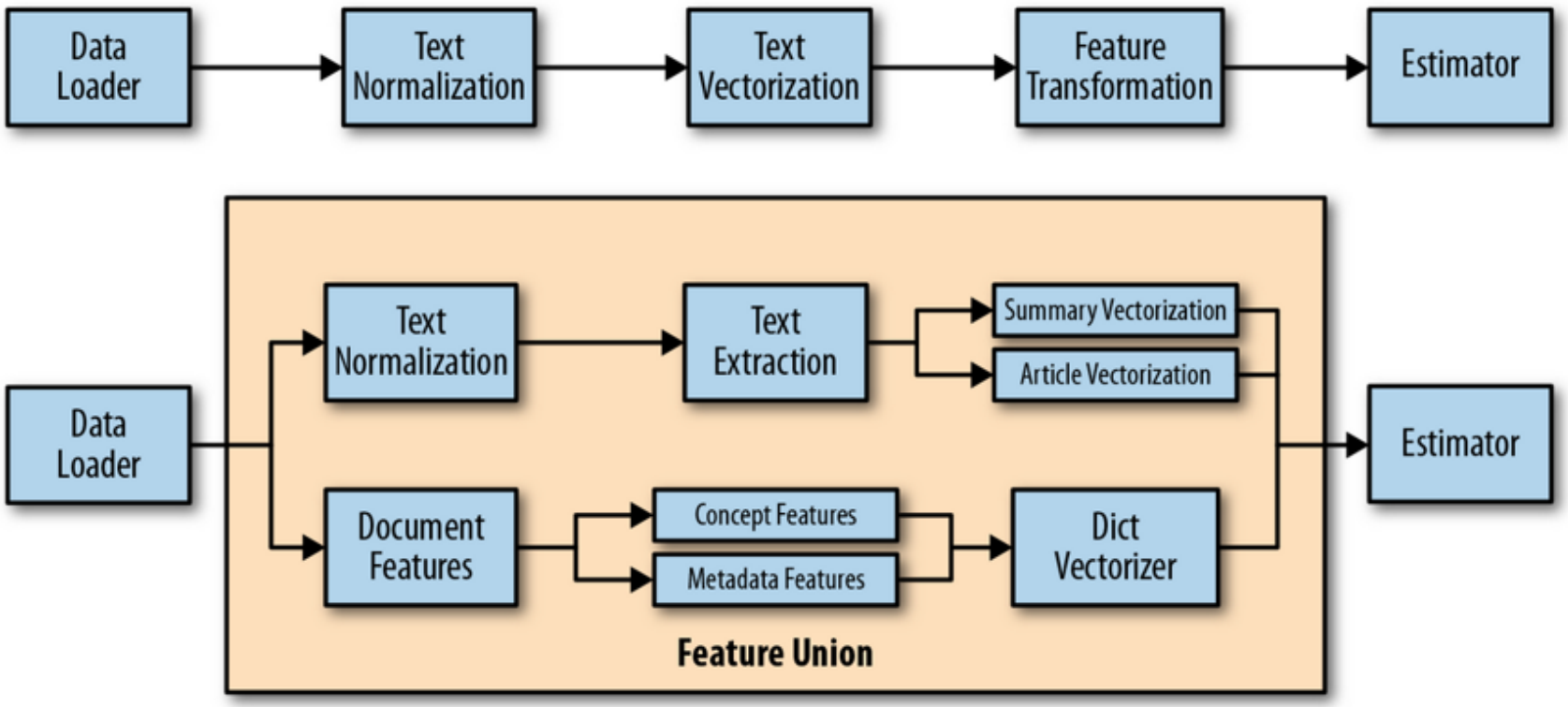
$$\text{Jaccard Similarity}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- **Word Embedding Similarity:**
  Word embeddings are typically represented as dense vectors. Similarity between two word vectors **A** and **B** can be calculated using cosine similarity or Euclidean distance, as shown earlier.
- **Document Embedding Similarity:**
  Document embeddings, represented as vectors, can also utilize cosine similarity or Euclidean distance for measuring similarity, as discussed previously.

These math formulas allow us to quantitatively assess the similarity or dissimilarity between vectors representing textual data in NLP tasks.

Text Preprocessing and Text Representation

Pipelines for text vectorization and feature extraction



https://www.oreilly.com/library/view/applied-text-analysis/9781491963036/ch04.html

- **Statistical language modeling** is the task of estimating the probability distribution of sequences of words in a language.
- A common approach is to use n-gram models, which assign probabilities to sequences of n words based on their frequency in a training corpus.
- Let $w_1, w_2, \ldots, w_n$ represent a sequence of words. The n-gram probability can be estimated using the chain rule of probability:

$$P(w_1, w_2, \ldots, w_n) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1, w_2) \cdot \ldots \cdot P(w_n|w_1, w_2, \ldots, w_{n-1})$$

Example

- Consider the sentence "I love to learn." We can estimate the probability of this sentence using a trigram model:

$$P(\text{"I love to learn."}) = P(\text{"I"}) \cdot P(\text{"love"}|\text{"I"}) \cdot P(\text{"to"}|\text{"I love"}) \cdot P(\text{"learn"}|\text{"I love to"})$$

Text Preprocessing and Text Representation

NLP preprocessing in Python - An Example of NLTK Library

In [2]:
```
import nltk
nltk.download('averaged_perceptron_tagger')
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk import pos_tag

# Sample text
text = "Natural language processing is a subfield of linguistics, computer science, and artificial intelligence concerned with th

# Tokenization
tokens = word_tokenize(text)

# Stop word removal
stop_words = set(stopwords.words('english'))
filtered_tokens = [token for token in tokens if token.lower() not in stop_words]

# Stemming
stemmer = PorterStemmer()
stemmed_tokens = [stemmer.stem(token) for token in filtered_tokens]

# Lemmatization
lemmatizer = WordNetLemmatizer()
lemmatized_tokens = [lemmatizer.lemmatize(token) for token in filtered_tokens]

# Part-of-speech tagging
pos_tokens = pos_tag(filtered_tokens)

print("Tokens:", tokens)
print("Filtered Tokens:", filtered_tokens)
print("Stemmed Tokens:", stemmed_tokens)
print("Lemmatized Tokens:", lemmatized_tokens)
print("POS Tagged Tokens:", pos_tokens)
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\atten\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
Tokens: ['Natural', 'language', 'processing', 'is', 'a', 'subfield', 'of', 'linguistics', ',', 'computer', 'science', ',', 'an
d', 'artificial', 'intelligence', 'concerned', 'with', 'the', 'interactions', 'between', 'computers', 'and', 'human', 'language
s', '.']
Filtered Tokens: ['Natural', 'language', 'processing', 'subfield', 'linguistics', ',', 'computer', 'science', ',', 'artificia
l', 'intelligence', 'concerned', 'interactions', 'computers', 'human', 'languages', '.']
Stemmed Tokens: ['natur', 'languag', 'process', 'subfield', 'linguist', ',', 'comput', 'scienc', ',', 'artifici', 'intellig',
'concern', 'interact', 'comput', 'human', 'languag', '.']
Lemmatized Tokens: ['Natural', 'language', 'processing', 'subfield', 'linguistics', ',', 'computer', 'science', ',', 'artificia
l', 'intelligence', 'concerned', 'interaction', 'computer', 'human', 'language', '.']
POS Tagged Tokens: [('Natural', 'JJ'), ('language', 'NN'), ('processing', 'NN'), ('subfield', 'NN'), ('linguistics', 'NNS'),
(',', ','), ('computer', 'NN'), ('science', 'NN'), (',', ','), ('artificial', 'JJ'), ('intelligence', 'NN'), ('concerned', 'VB
N'), ('interactions', 'NNS'), ('computers', 'NNS'), ('human', 'JJ'), ('languages', 'NNS'), ('.', '.')]
```

https://medium.com/datadriveninvestor/a-guide-to-natural-language-processing-basics-d30c69bbe816

Supervised Learning for NLP

Classification Tasks in NLP

Classification tasks in Natural Language Processing (NLP) involve assigning predefined labels or categories to textual data. These tasks play a crucial role in various applications. Let's explore some common classification tasks in NLP:

1. **Sentiment Analysis:** Sentiment analysis aims to determine the sentiment or opinion expressed in a text. It can involve classifying text as positive, negative, or neutral, or assigning sentiment scores to indicate the intensity of sentiment.
   Example: Classifying customer reviews as positive or negative.
2. **Text Categorization:** Text categorization involves assigning documents to predefined categories or topics. It can be used for document classification, news categorization, spam detection, etc.
   Example: Classifying news articles into topics such as sports, politics, or entertainment.
3. **Topic Detection:** Topic detection focuses on identifying the main topics or themes present in a collection of documents. It can involve unsupervised techniques such as clustering or supervised techniques with predefined topic labels.
   Example: Detecting topics in social media conversations or research papers.
4. **Intent Classification:** Intent classification aims to determine the intent or purpose behind a user's text query or command. It is commonly used in chatbots, virtual assistants, and customer support systems.
   Example: Classifying user queries as inquiries, bookings, or complaints.
5. **Named Entity Recognition (NER):** NER involves identifying and classifying named entities such as persons, organizations, locations, and dates in text. It is useful for information extraction and knowledge base construction.
   Example: Identifying names of people, companies, and locations in news articles.

Supervised Learning for NLP

AI Methods for Classification Tasks in NLP

Classification tasks in Natural Language Processing (NLP) involve assigning predefined labels or categories to textual data. These tasks are essential in various applications, and AI methods play a crucial role in achieving accurate and efficient classification. Let's explore some AI methods commonly used for classification tasks in NLP:

1. **Naive Bayes:** Naive Bayes classifiers are probabilistic models that utilize Bayes' theorem to make predictions. They assume independence between features and calculate the conditional probability of each class given the features. Naive Bayes is widely used in text classification tasks, such as sentiment analysis and spam detection.
   Example: Classifying movie reviews as positive or negative based on the words used in the text.
2. **Support Vector Machines (SVM):** SVM is a powerful machine learning algorithm that finds an optimal hyperplane to separate data into different classes. SVMs can be applied to NLP tasks by representing text data as feature vectors. They are effective in text categorization, sentiment analysis, and named entity recognition.
   Example: Categorizing news articles into topics such as sports, politics, or entertainment based on their textual content.
3. **Neural Networks:** Neural networks, particularly deep learning models, have shown remarkable success in NLP. Architectures like Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) can capture sequential and contextual information in text, leading to improved classification performance.
   Example: Intent classification in chatbots, where the neural network predicts the user's intent (e.g., inquiry, booking, complaint) based on their text query.

- **Text classification** aims to assign predefined categories or labels to text documents.
- One popular statistical method for text classification is the Naive Bayes classifier.
- Given a document d and a set of classes $C = \{c_1, c_2, \ldots, c_k\}$, the Naive Bayes classifier predicts the class with the highest posterior probability using Bayes' theorem:

$$\hat{c} = \arg\max_{c \in C} P(c|d) = \arg\max_{c \in C} \frac{P(d|c) \cdot P(c)}{P(d)}$$

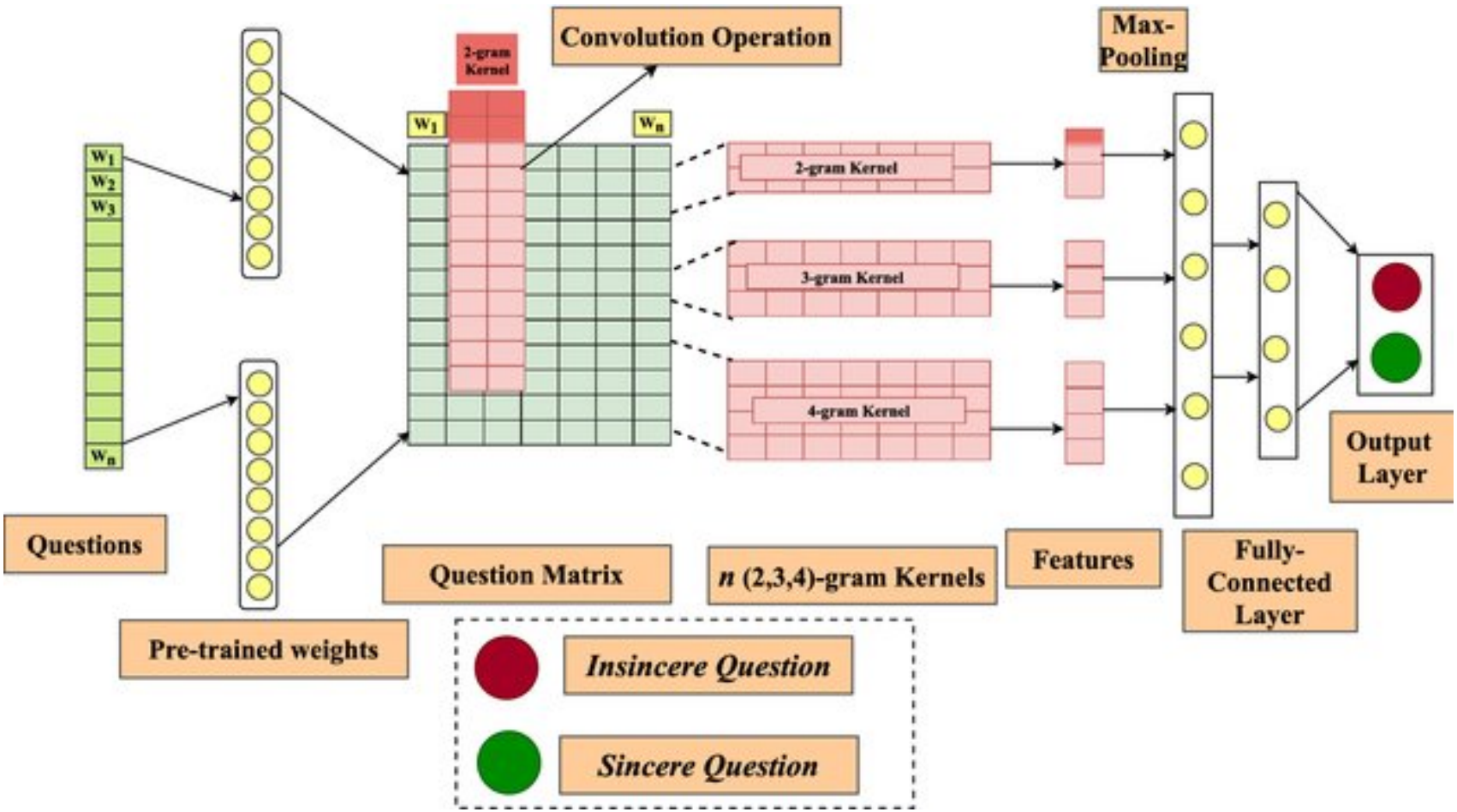
Example

- Suppose we have two classes: "sports" and "politics". Given a document d, we can predict the class using the Naive Bayes classifier.

$$\hat{c} = \arg\max_{c \in \{\text{"sports"}, \text{"politics"}\}} \frac{P(d|\text{c}) \cdot P(\text{c})}{P(d)}$$

Supervised Learning for NLP

CNN for text classification



https://www.researchgate.net/publication/342348947

Unsupervised Learning for NLP

Clustering Techniques in NLP

Clustering techniques in Natural Language Processing (NLP) group similar documents or text data into clusters based on their intrinsic characteristics. These techniques are useful for tasks such as document organization, topic modeling, and information retrieval.

1. **K-means Clustering:** K-means is a widely used clustering algorithm that partitions data into k clusters, where k is predefined. It iteratively assigns data points to the nearest centroid, aiming to minimize the sum of squared distances. K-means clustering can be applied to NLP by representing text data as numerical feature vectors.
   Example: Clustering news articles into different groups based on their content to identify distinct topics.
2. **Hierarchical Clustering:** Hierarchical clustering builds a hierarchy of clusters using either agglomerative (bottom-up) or divisive (top-down) approaches. It merges or splits clusters based on their similarity until a stopping criterion is met. Hierarchical clustering can capture nested structures and allow exploration of different levels of granularity.
   Example: Grouping tweets into clusters based on the similarity of their content to identify related discussions.
3. **Density-based Clustering:** Density-based clustering algorithms, such as DBSCAN (Density-Based Spatial Clustering of Applications with Noise), identify dense regions in the data and group data points based on their connectivity. Unlike partitioning algorithms, density-based methods can discover clusters of arbitrary shape and handle noise and outliers effectively.
   Example: Clustering customer reviews to identify distinct opinions and sentiments about a product or service.

Topic modeling is a statistical modeling approach that automatically identifies topics or themes present in a collection of documents. It assigns each document a probability distribution over a set of predefined topics and represents topics as distributions over words.

- **Latent Dirichlet Allocation (LDA):** LDA is one of the most widely used topic modeling algorithms. It assumes that each document is a mixture of various topics, and each word in the document is generated by selecting a topic and then selecting a word from that topic's word distribution. LDA uses probabilistic inference to estimate the topic distribution and word distributions.
- **Non-negative Matrix Factorization (NMF):** NMF is another popular topic modeling technique. It represents documents and words as non-negative matrices and factorizes the document-term matrix into two lower-rank matrices representing document-topic and topic-word distributions. NMF discovers latent topics by minimizing the reconstruction error.
- **Probabilistic Latent Semantic Analysis (pLSA)**
- **Hierarchical Dirichlet Process (HDP)**
- **...**

- **Applications:** Topic modeling has a wide range of applications, including:
  - Document Organization: Grouping similar documents together based on their topics, facilitating efficient information retrieval and organization.
  - Recommendation Systems: Identifying topics of interest for users based on their interactions with documents, allowing personalized recommendations.
  - Text Summarization: Extracting key topics and generating concise summaries of large volumes of text.
- **Evaluation Metrics:** Evaluating the quality of topic models is essential.
  - Common evaluation metrics include coherence measures, such as topic coherence or topic diversity, which assess the interpretability and coherence of the generated topics.

Introduction

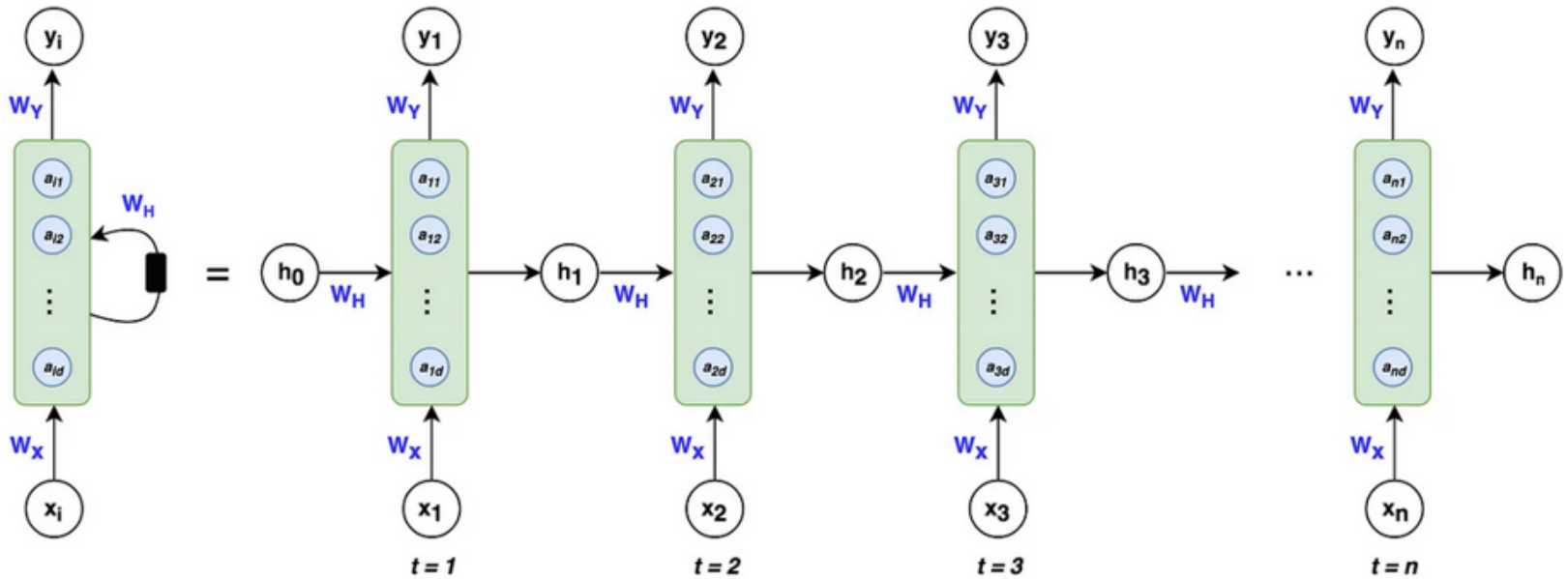
- Deep Learning methods have significantly advanced Natural Language Processing (NLP) by leveraging neural network architectures to model complex linguistic patterns.
- In this presentation, we will explore various Deep Learning techniques used in NLP, including Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and Transformer-based models.
- These methods have demonstrated state-of-the-art performance in tasks such as machine translation, sentiment analysis, and named entity recognition.

Recurrent Neural Networks (RNNs)

- RNNs are widely used in NLP for modeling sequential data, capturing temporal dependencies.
- The hidden state h_t at time step t is computed as a function of the input x_t and the previous hidden state h_{t-1}:

$$h_t = \mathrm{f}(W_{hh}h_{t-1} + W_{xh}x_t)$$

  where W_{hh} and W_{xh} are weight matrices, and f is an activation function.
- Example: Language Modeling
  - Language modeling involves predicting the next word in a sequence given previous words.
  - RNNs, such as LSTM or GRU, are capable of capturing long-term dependencies and have achieved impressive language modeling performance.
  - By training an RNN on a large text corpus, it can generate coherent and contextually appropriate text.



Left: Shorthand notation often used for RNNs, **Right**: Unfolded notation for RNNs

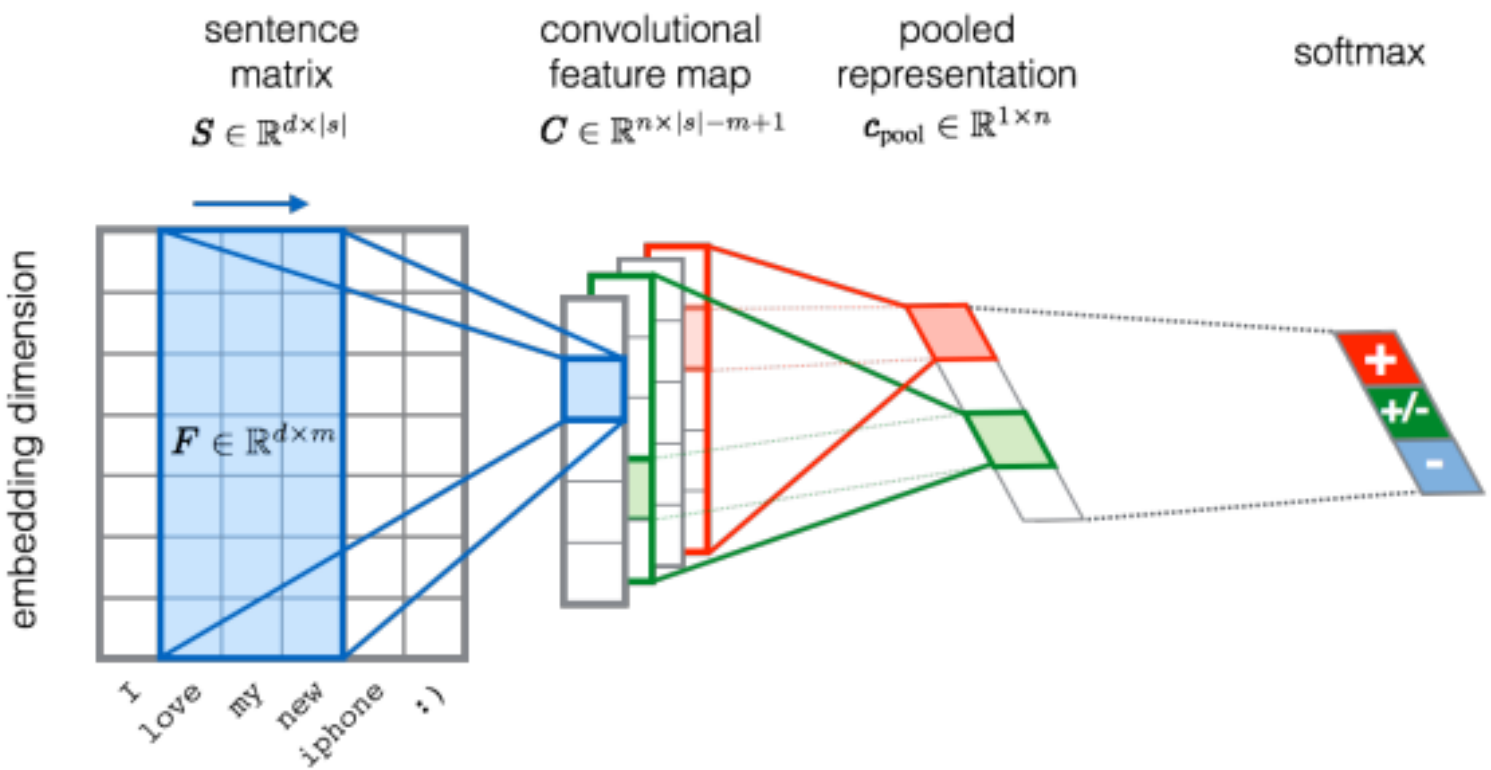
https://ai.stackexchange.com/questions/4683/what-is-the-fundamental-difference-between-cnn-and-rnn

Convolutional Neural Networks (CNNs)

- CNNs, originally designed for computer vision, have been successfully applied to NLP tasks.
- CNNs use convolutional layers to extract local features from input sequences.
- Example: Text Classification
  - Text classification tasks, such as sentiment analysis or topic categorization, can benefit from CNNs.
  - By treating text as a 1D signal, CNNs can capture important local patterns or n-gram features.
  - Multiple convolutional filters with different kernel sizes can learn diverse features, which are then combined and fed into a classifier for prediction.

Deep Learning Methods for NLP

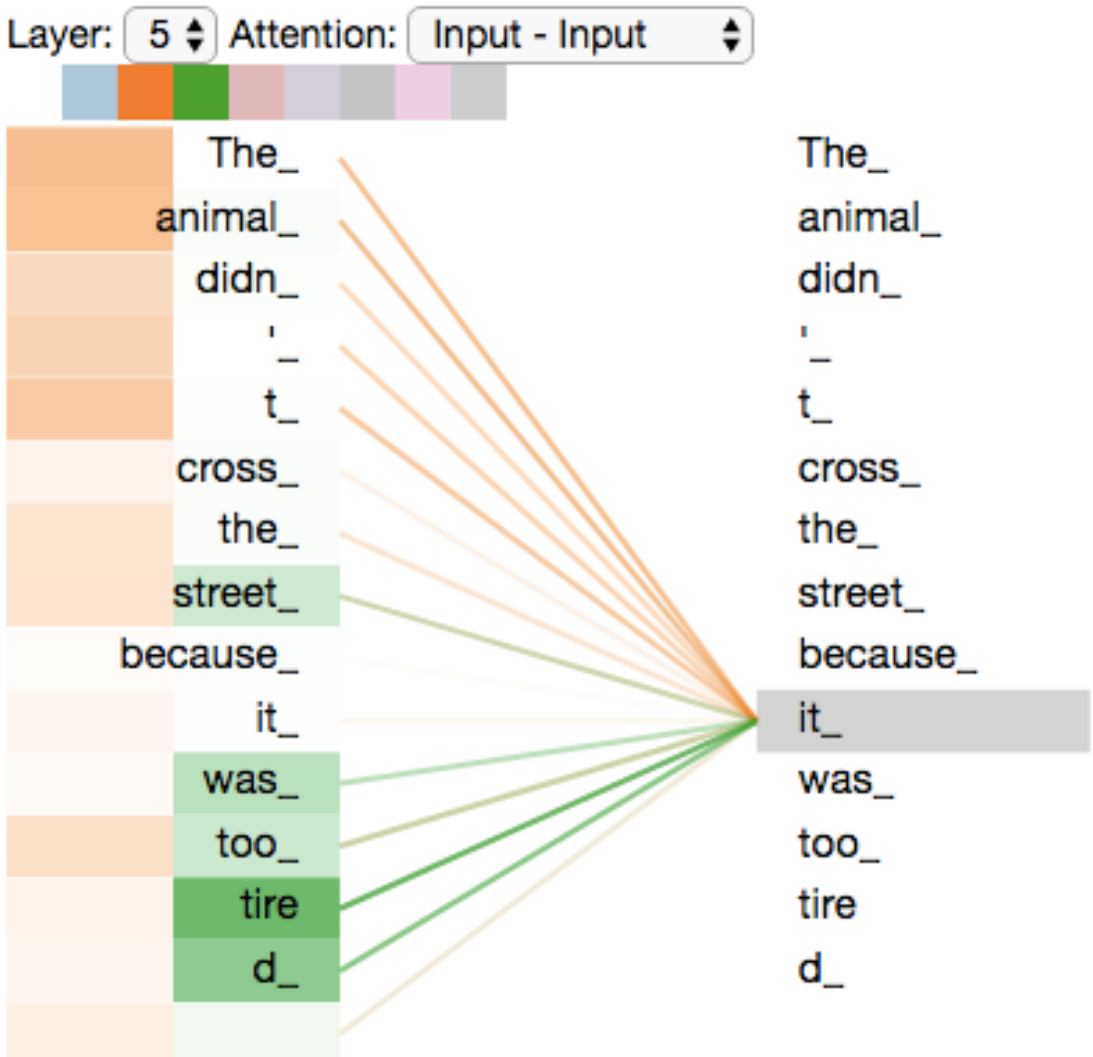
CNN High-level Architecture



https://towardsdatascience.com/convolutional-neural-network-in-natural-language-processing-96d67f91275c

Attention-based Models

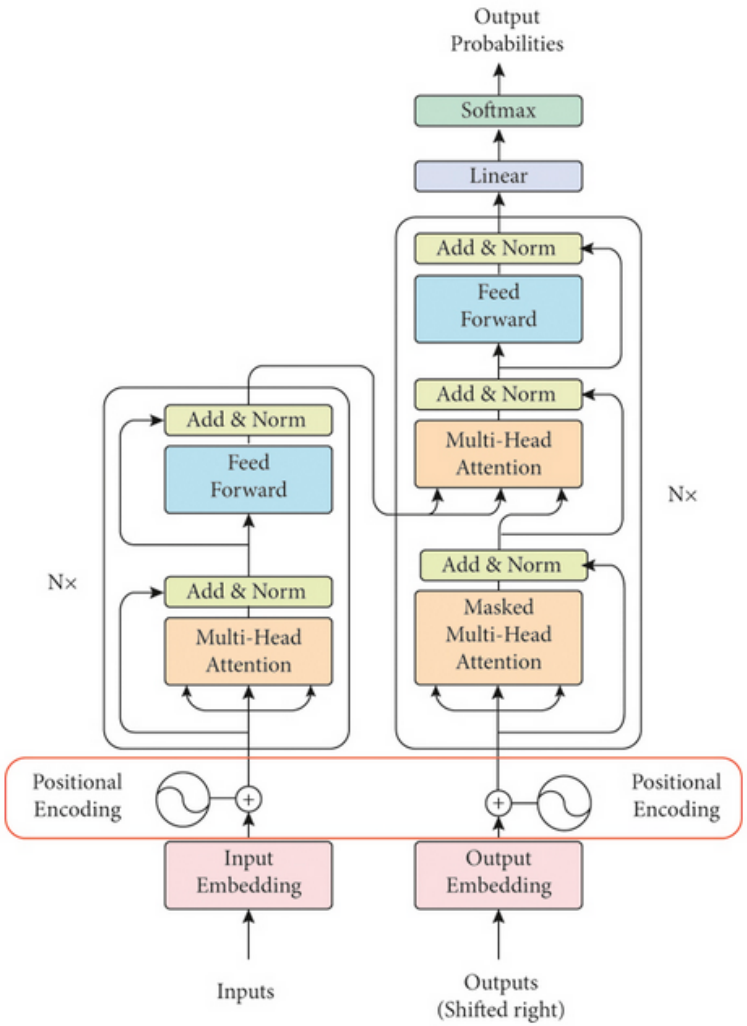
- Attention-based models have revolutionized NLP and achieved state-of-the-art results in various tasks.
- These models utilize attention mechanisms to capture global dependencies between words in a sequence.
- By dynamically attending to relevant words, they can effectively capture contextual information and improve performance.
- Example: Machine Translation
  - Machine translation involves converting text from one language to another.
  - Traditional approaches faced challenges in capturing complex linguistic patterns and handling word reordering in translations.
  - However, attention-based models have demonstrated remarkable performance in this task.
  - They can focus on relevant words across the entire sequence, allowing for better translation quality.
  - Attention mechanisms assign higher weights to important words, enabling the model to capture relevant information for translation.



https://towardsdatascience.com/what-is-attention-mechanism-can-i-have-your-attention-please-3333637f2eac

Deep Learning Methods for NLP

NN Transformer Architecture



https://www.hindawi.com/journals/cin/2022/1883698/

Transformers - Self-Attention

- Transformers employ self-attention mechanisms to capture relationships between different words in a sequence.
- Self-attention allows each word to attend to other words in the same sequence, capturing their importance or relevance.
- Given an input sequence of word embeddings $x_1, x_2, \ldots, x_n$, the self-attention mechanism computes a weighted sum of values based on the similarity between a query and a set of key-value pairs.
- Mathematically, the self-attention mechanism is defined as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where Q, K, and V are query, key, and value matrices, respectively, and d_k is the dimension of the key vectors.

Transformers - Multi-Head Attention

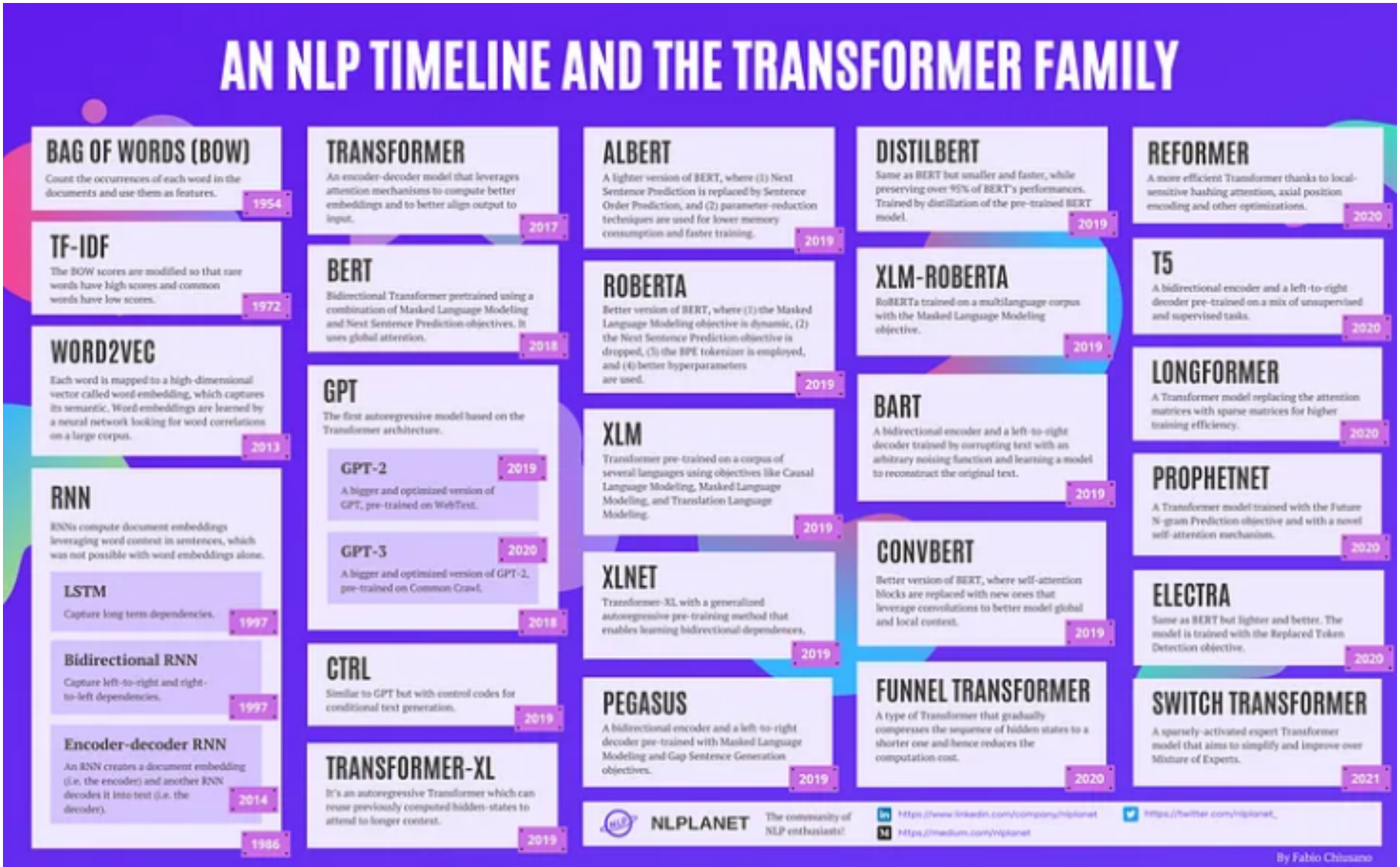
- Transformers use multi-head attention to capture different types of information and increase model capacity.
- Multi-head attention splits the query, key, and value vectors into multiple parallel sets, each with its own projection matrices.
- These parallel attention heads capture different dependencies and learn diverse representations.
- The outputs of all attention heads are concatenated and linearly transformed to obtain the final attention output.
- Mathematically, the multi-head attention mechanism is defined as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$ is the i-th attention head, and W^O is the output projection matrix.

Deep Learning Methods for NLP

Landscape of Transformers for NLP



https://medium.com/nlplanet/a-brief-timeline-of-nlp-from-bag-of-words-to-the-transformer-family-7caad8bbba56

Challenges in NLP and AI

- Ambiguity in human language and natural language processing
- Diversity in human language across different regions and cultures
- Lack of training data for specific domains and applications
- Ethical considerations and biases in AI models and applications

Solutions to NLP and AI Challenges

- Development of advanced NLP models with enhanced natural language understanding capabilities
- Building multilingual NLP models for cross-lingual applications
- Data augmentation and transfer learning techniques for domain-specific applications
- Ethical guidelines and bias mitigation strategies for responsible AI development

Natural Language Processing and Artificial Intelligence

Summary

1. S. J. Russell, P. Norvig, Artificial Intelligence: A Modern Approach", Financial Times Prentice Hall, 2019.
2. H. Lane, C. Howard, H.M. Hapke, "Natural Language Processing in Action", Manning Publication Co., 2019
3. M. Flasiński, Introduction to Artificial Intelligence", Springer Verlang, 2016