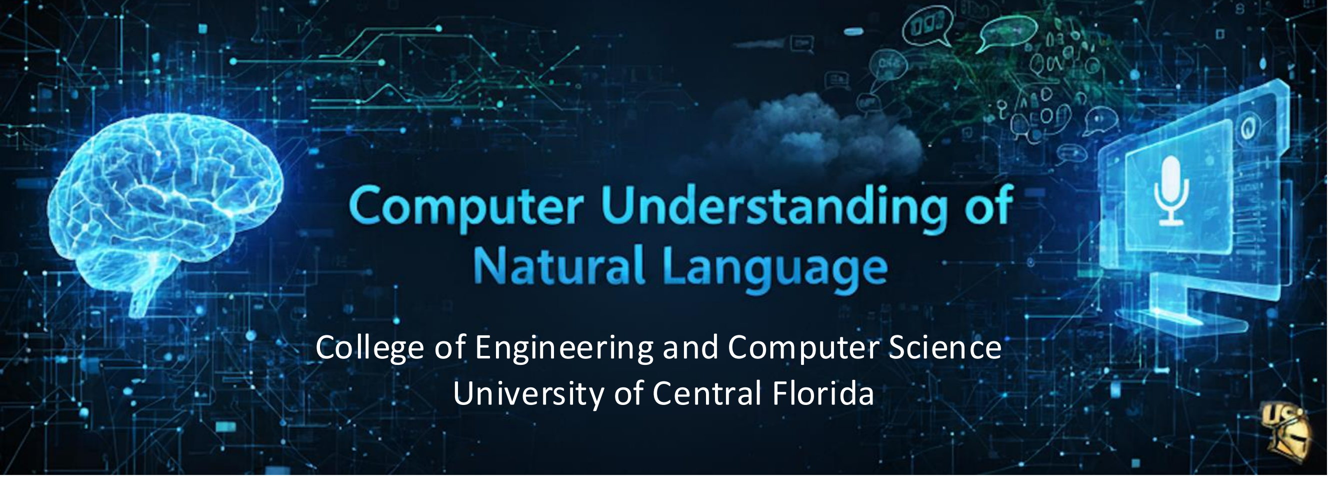


David Mohaisen, Professor
University of Central Florida
Department of Computer Science
Cybersecurity and Privacy Cluster

https://cs.ucf.edu/~mohaisen/

Historical Evolution and Core Tasks in NLP

① Historical background
② Evolution
③ Why NLP is hard
④ NLP tasks
⑤ Computer vs. natural languages
⑥ Usable meanings

Historical Background (1/2)

- Mid 1950s-1960s: birth of NLP and linguistics
  - Initially thought to be easy
  - Mostly relied on hand-coded rules

- Mid 1960s-1970s: dark times
  - After initial hype, dark era followed
  - People started believing machine transaction is impossible
  - Most abandoned NLP research

Historical Background (2/2)

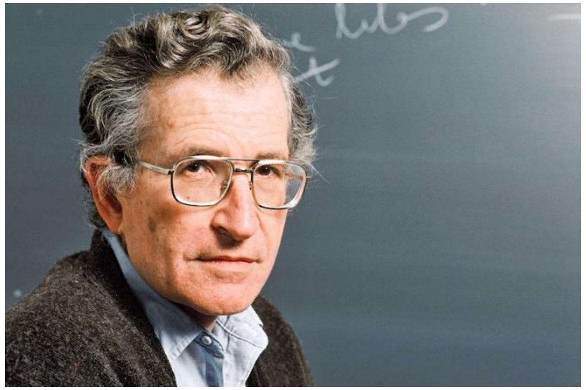
- Mid 1970s-1980s: slow revival:
  - Some efforts, mainly focused on linguistics
- Late 1980s-1990s:
  - A revolution with statistical modeling
  - Computing power increased substantially
  - Data-driven, statistical models with simple representations beat complex hand-code rules
- 2000's: statistics powered by linguistics
  - Better results with more sophisticated statistical analysis models and tools
- 2010's: representations are an automatic drive using learning techniques that combine statistical, linguistic, and context rules.

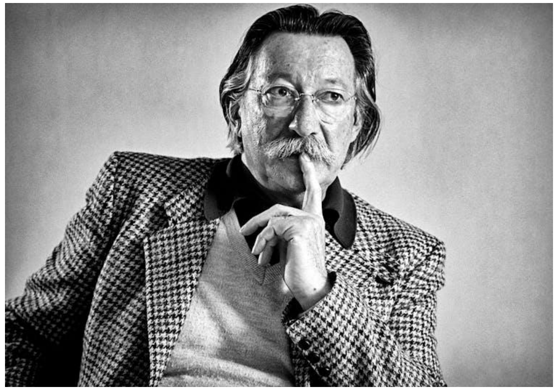
Historical Evolution (1/6)

- **Early Beginnings:**
  - **1950s:** Alan Turing proposed the concept of machine intelligence in his seminal paper, "Computing Machinery and Intelligence," introducing the **Turing Test** to evaluate a machine's ability to exhibit intelligent behavior.
  - **1954:** The **Georgetown-IBM experiment** demonstrated the first machine translation system, translating 60 Russian sentences into English, marking an early milestone in NLP.

Historical Evolution (2/6)

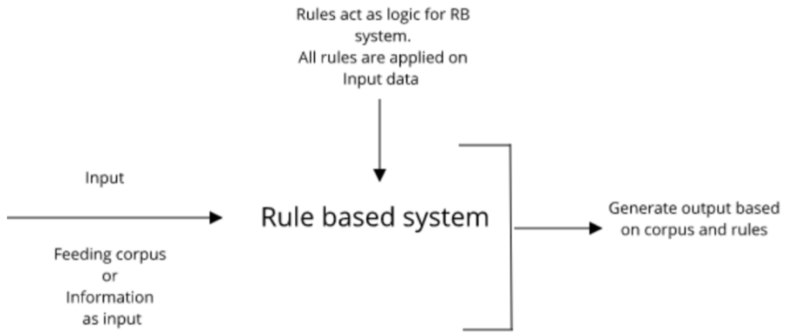
- **Rule-Based Era:**
  - **1960s-1980s:** NLP was dominated by symbolic systems using hand-crafted rules and grammars.
    - **Noam Chomsky's transformational grammar** (1957) influenced linguistic models, enabling rule-based parsing and analysis of language structures.
    - **Joseph Weizenbaum** created **ELIZA** (1966), one of the first chatbot programs, simulating a Rogerian psychotherapist.

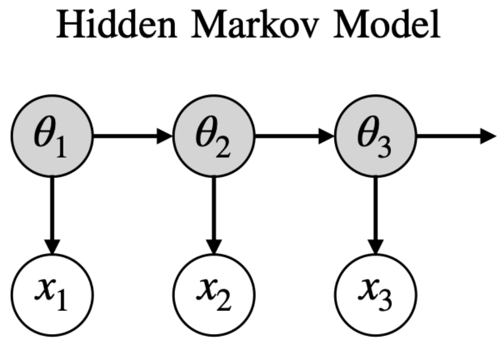




Historical Evolution (3/6)

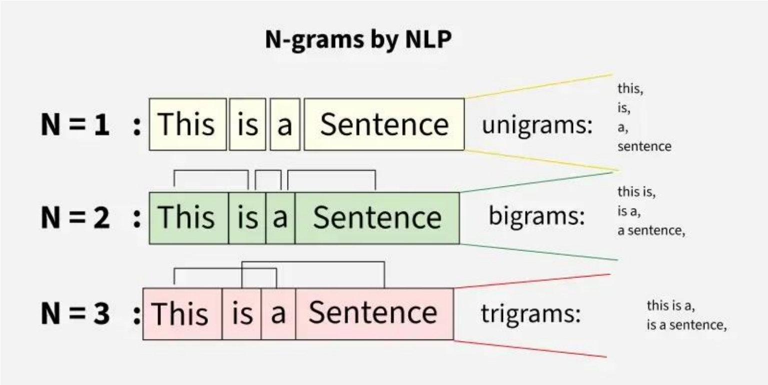
- **Early Machine Learning:**
  - **1980s-1990s:** Statistical methods began to replace rule-based systems as researchers gained access to larger datasets and computational power.
    - The introduction of **Hidden Markov Models (HMMs)** for speech and text processing revolutionized the field.
    - The Penn Treebank dataset (1993), created by **Mitchell Marcus**, provided annotated text corpora for training models.





Historical Evolution (4/6)

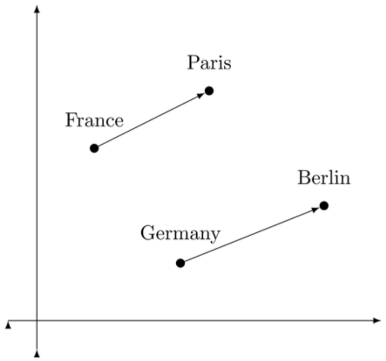
- **Statistical Revolution:**
  - **1990s-2000s:** Statistical NLP flourished, with algorithms like **n-grams** and **log-linear models** improving tasks like machine translation and speech recognition.
    - **BLEU Score (2002):** Developed by **Kishore Papineni**, it became the standard evaluation metric for machine translation.
    - Notable advancements included **Google Translate (2006)** leveraging statistical translation methods.

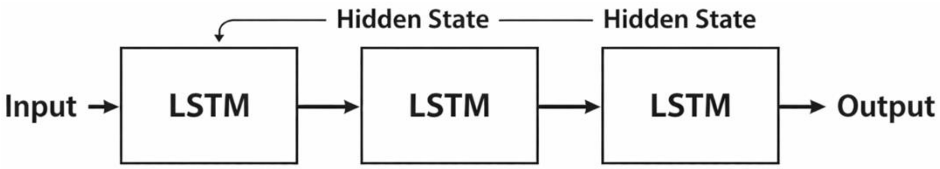


$$\mathrm{BLEU} = \min\left(1, \frac{\text{output-length}}{\text{reference-length}}\right)\left(\prod_{i=1}^{4} \text{precision}_i\right)^{\frac{1}{4}}$$

Historical Evolution (5/6)

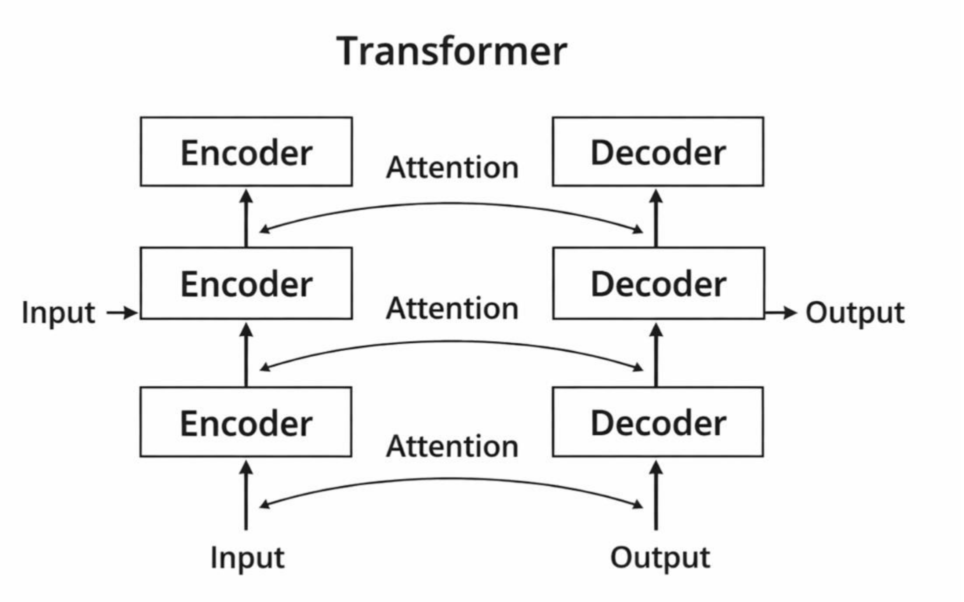
- **Deep Learning Era:**
  - **2010s:** Neural networks revolutionized NLP, focusing on representation learning and context understanding.
    - **Word Embeddings: Tomas Mikolov** introduced **Word2Vec (2013)**, transforming how semantic relationships between words are modeled.
    - **Sequence Models: Long Short-Term Memory (LSTM)** networks, proposed by **Hochreiter and Schmidhuber (1997)**, gained widespread adoption for sequential tasks like language modeling.

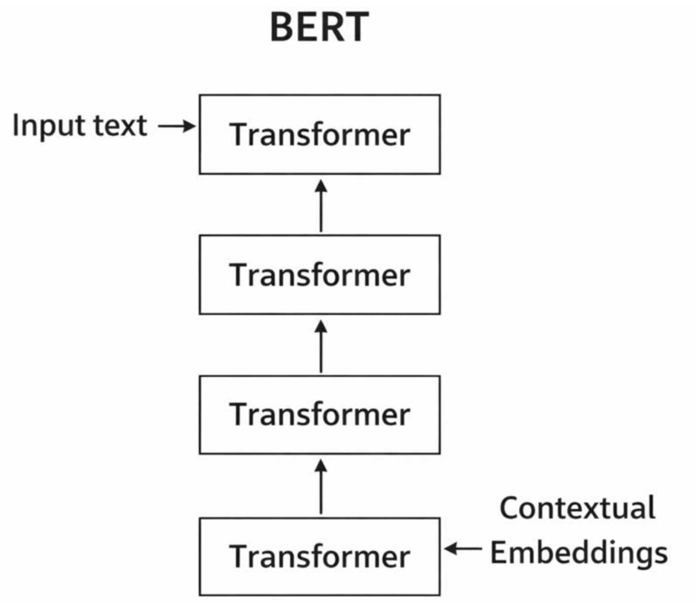




Historical Evolution (6/6)

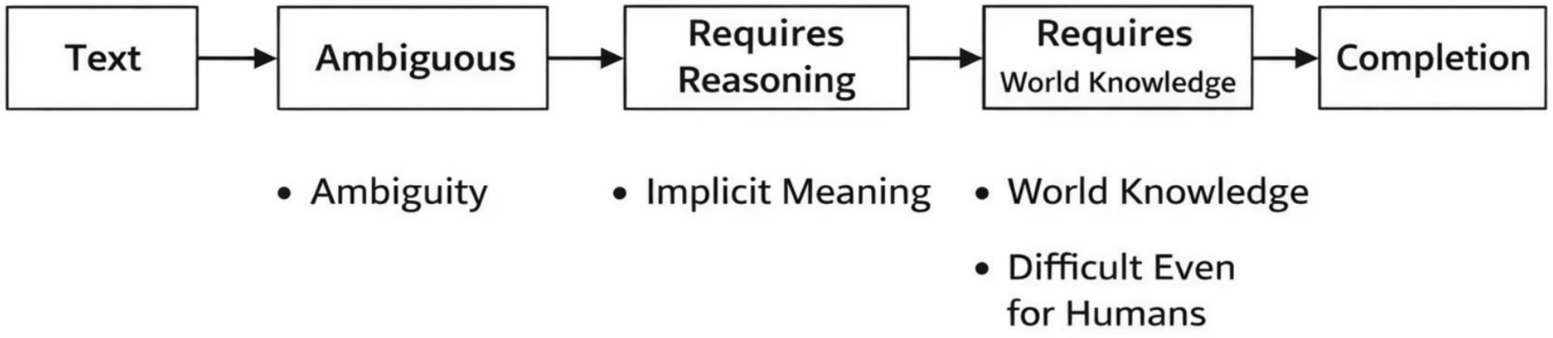
- **Transformer Models and Modern NLP:**
  - **2017:** The introduction of the **Transformer architecture** by **Vaswani et al.** paved the way for state-of-the-art models like **BERT (2018)** by **Google AI** and **GPT (2018)** by **OpenAI**.
    - Transformers enabled breakthroughs in contextual understanding and multitask learning, powering advanced systems like **ChatGPT** and **Bard**.





Why is NLP Hard?

- Human language is ambiguous
- Language requires reasoning beyond explicitly stated information
    - Often requires world knowledge
- Language is even difficult for humans



Why is NLP Hard? Ambiguity

- Ambiguities compound to generate enormous numbers of possible interpretations.
- In English, a sentence ending in n prepositional phrases has over 2^n syntactic interpretations.
  - "I saw the man with the telescope": 2 parses
  - "I saw the man on the hill with the telescope.": 5 parses
  - "I saw the man on the hill in Texas with the telescope": 14 parses
  - "I saw the man on the hill in Texas with the telescope at noon.": 42 parses
  - "I saw the man on the hill in Texas with the telescope at noon on Monday" 132 parses
  - .. And so forth

NLP Tasks and Their Types

- **Syntactic Tasks:** Focus on the structure and grammar of language.
  - **Examples:** Part-of-Speech Tagging, Dependency Parsing, Sentence Tokenization.
  - **Applications:** Grammar checkers, text preprocessing.

- **Semantic Tasks:** Focus on meaning and interpretation.
  - **Examples:** Named Entity Recognition (NER), Sentiment Analysis, Text Summarization.
  - **Applications:** Chatbots, search engines, recommendation systems.

- **Pragmatic Tasks:** Focus on context, intent, and implied meaning.
  - **Examples:** Coreference Resolution, Dialogue Management, Discourse Analysis.
  - **Applications:** Virtual assistants, conversational AI, customer support.

Syntactic: Understanding Language Structure

- Focus on analyzing the grammatical structure of language to ensure correctness and coherence.

- Examples include:
  - **Part-of-Speech (POS) Tagging:** Identifying the grammatical category of each word (e.g., noun, verb, adjective).
  - **Dependency Parsing:** Mapping the syntactic structure to identify relationships between words (e.g., subject-verb-object relations).
  - **Sentence Splitting and Tokenization (segmentation):** Breaking down text into sentences and words for further analysis.

- **Applications:** Grammar checkers (e.g., Grammarly), sentence restructuring, and preprocessing for downstream NLP tasks.

Syntactic: Word Segmentation

- Breaking a string of characters into a sequence of words.
- In some written languages (e.g. Chinese) words are not separated by spaces.
- Even in English, characters other than white-space can be used to separate words [e.g. , ; . - : ( )]
- Examples from English URLs:
  - jumptheshark.com => jump the shark .com
  - myspace.com/pluckerswingbar
    - myspace .com pluckers wing bar
    - myspace .com plucker swing bar

Syntactic: Morphological Analysis

- *Morphology* is the field of linguistics that studies the internal structure of words.
- A *morpheme* is the smallest linguistic unit that has semantic meaning
  - e.g. “carry”, “pre”, “ed”, “ly”, “s”
- *Morphological* analysis is the task of segmenting a word into its morphemes:
  - carried => carry + ed (past tense)
  - independently => in + (depend + ent) + ly
  - Googlers => (Google + er) + s (plural)
  - unlockable => un + (lock + able) ?
    - => (un + lock) + able ?

Syntactic: Part of Speech Tagging

- Annotate each word in a sentence with part of speech tag
  - I ate the pasta with meatballs
  - I saw the saw and decide to take it to the table
- Useful for subsequent syntactic parsing and word sense disambiguation.

Semantic: Understanding Meaning

- Aim to capture the meaning of words, sentences, or entire texts, going beyond surface-level structure. Examples include:
  - **Named Entity Recognition (NER):** Identifying proper nouns like people, places, organizations, etc.
  - **Word Sense Disambiguation:** Determining the correct meaning of a word in context (e.g., "bank" as a financial institution vs. a riverbank).
  - **Text Summarization:** Generating concise and meaningful summaries of longer texts.
  - **Sentiment Analysis:** Detecting the emotional tone or sentiment in text (e.g., positive, negative, neutral).
  - **Applications:** Chatbots, search engines, recommendation systems, and social media monitoring.

Semantic: Word Disambiguation

- Words in natural language usually have a fair number of different possible meanings.
  - Ellen has a strong interest in computational linguistics.
  - Ellen pays a large amount of interest on her credit card.
- For many tasks (question answering, translation), the proper sense of each ambiguous word in a sentence must be determined.

Pragmatic: Understanding Context and Intent

- Deal with interpreting language in context, accounting for situational factors, tone, and implied meanings. Examples include:
  - **Coreference Resolution:** Linking pronouns or references back to the entities they refer to (e.g., "John went to the store. He bought milk.").
  - **Dialogue Management:** Understanding conversational context to maintain coherent interactions in chatbots and virtual assistants.
  - **Speech Act Recognition:** Identifying the purpose of a statement (e.g., a question, command, or assertion).
  - **Discourse Analysis:** Analyzing how sentences connect to form coherent text.
  - **Applications:** Virtual assistants (e.g., Siri, Alexa), customer support systems, and personalized learning platforms.

Pragmatic: Coreference

- Determine which phrases in a document refer to the same underlying entity.
  - *John* put the *carrot* on the *plate* and ate *it*.
  - Bush started the war in Iraq. But the president needed the consent of Congress.

- Some cases require difficult reasoning.
  - Today was Jack's birthday. Penny and Janet went to the store. They were going to get presents. Janet decided to get *a kite*. "Don't do that," said Penny. "Jack has *a kite*. He will make you take *it* back."

Takeaway

By combining syntactic, semantic, and pragmatic understanding, NLP systems can analyze, comprehend, and respond to human language effectively, enabling smarter, context-aware applications.

Computer vs. Natural Languages

- Ambiguity is the primary difference between natural and computer languages.
- Formal programming languages are designed to be unambiguous, i.e. they can be defined by a grammar that produces a unique parse for each sentence in the language.
- Programming languages are also designed for efficient (deterministic) parsing.

(1) Ambiguity in Natural Languages

- Natural languages, like English, Arabic, or Spanish, are inherently **ambiguous**.
  - A single word or sentence can have multiple interpretations depending on **context, tone, or cultural nuances**.
  - **Example:** "The chicken is ready to eat."
    - Ambiguous: Is the chicken eating or being eaten?
- Ambiguity arises due to factors such as:
  - Polysemy: Words with multiple meanings (e.g., "bank" as a financial institution vs. a riverbank).
  - Syntax: Different grammatical structures leading to different meanings.
  - Pragmatics: Intent and context influencing interpretation.

(2) Unambiguity in Programming Languages:

- **Formal programming languages** (e.g., Python, Java, C++) are explicitly designed to be **unambiguous**:
  - Every statement in the language has a **clear, unique interpretation**.
  - This is achieved through a well-defined **grammar** (e.g., Backus-Naur Form) that precisely dictates syntax and structure.
  - **Example:** The statement x = x + 1; always updates the value of x deterministically.

(3) Efficient and Deterministic Parsing in Programming Languages:

- Programming languages are designed for efficient **parsing and execution**:
  - **Parsing:** The process of analyzing code structure to ensure correctness (using parsers like LL or LR parsers).
    - Parsing in formal languages is **deterministic**, meaning there is a single, predictable way to interpret the code.
  - **Efficiency:** Ensures that compilers and interpreters can quickly translate code into machine-executable instructions.

Usable Meanings in Computers

- Common solution (WordNet): use a thesaurus containing lists of synonyms sets and hypernyms (“is a” relationships)

e.g. synonym sets containing “good”:

```
from nltk.corpus import wordnet as wn
poses = { 'n':'noun', 'v':'verb', 's':'adj (s)', 'a':'adj', 'r':'adv'}
for synset in wn.synsets("good"):
    print("{}: {}".format(poses[synset.pos()],
            ", ".join([l.name() for l in synset.lemmas()])))
```

```
noun: good
noun: good, goodness
noun: good, goodness
noun: commodity, trade_good, good
adj: good
adj (sat): full, good
adj: good
adj (sat): estimable, good, honorable, respectable
adj (sat): beneficial, good
adj (sat): good
adj (sat): good, just, upright
…
adverb: well, good
adverb: thoroughly, soundly, good
```

e.g. hypernyms of “panda”:

```
from nltk.corpus import wordnet as wn
panda = wn.synset("panda.n.01")
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

Problems with WordNet

- Great as a resource but missing nuance
  - “proficient” is listed as a synonym for “good”.
  - This is only correct in some contexts.
- Missing new meanings of words:
  - wicked, badass, nifty, wizard, genius, ninja, bombast, …; a lot of new words
  - Impossible to keep up-to-date without effort
- Varying degrees of subjectivity
- Requires human labor to create and adapt
- Cannot compute accurate word similarity