MODULE 1 • CS24351

Module 1: Introduction to NLP, Linguistic Levels & Preprocessing

Topics 1 to 12 • Phonology, Morphology, Syntax, Ambiguity, Tokenization, Normalization & Edit Distance

Module 1: Introduction to NLP, Linguistic Levels & Text Preprocessing

📌 **Syllabus & Pedagogical Blueprint (Topics 1 to 12)**

Foundations of Computational Linguistics • Scientific vs. Engineering Goals • The End-to-End NLP Pipeline • Comprehensive Taxonomy of Linguistic Ambiguities • The 6 Levels of Language Analysis (Phonetics/Phonology, Morphology, Syntax, Semantics, Pragmatics, Discourse) • Tokenization Mechanics (Word, Sentence, Subword BPE, WordPiece, SentencePiece) • Normalization Pipelines (Case Folding, Unicode Decomposition NFKD, Levenshtein Minimum Edit Distance with Dynamic Programming) • Stemming vs. Lemmatization (Porter Stemmer Rules 1a–5b, Snowball, WordNet Lemmatizer) • Stop-Word Filtering • Regular Expressions, Finite State Automata (NFA/DFA), and Regex-Based Entity Extraction • 15 Solved University Examination Questions.

Topic 1: Foundations of Natural Language Processing

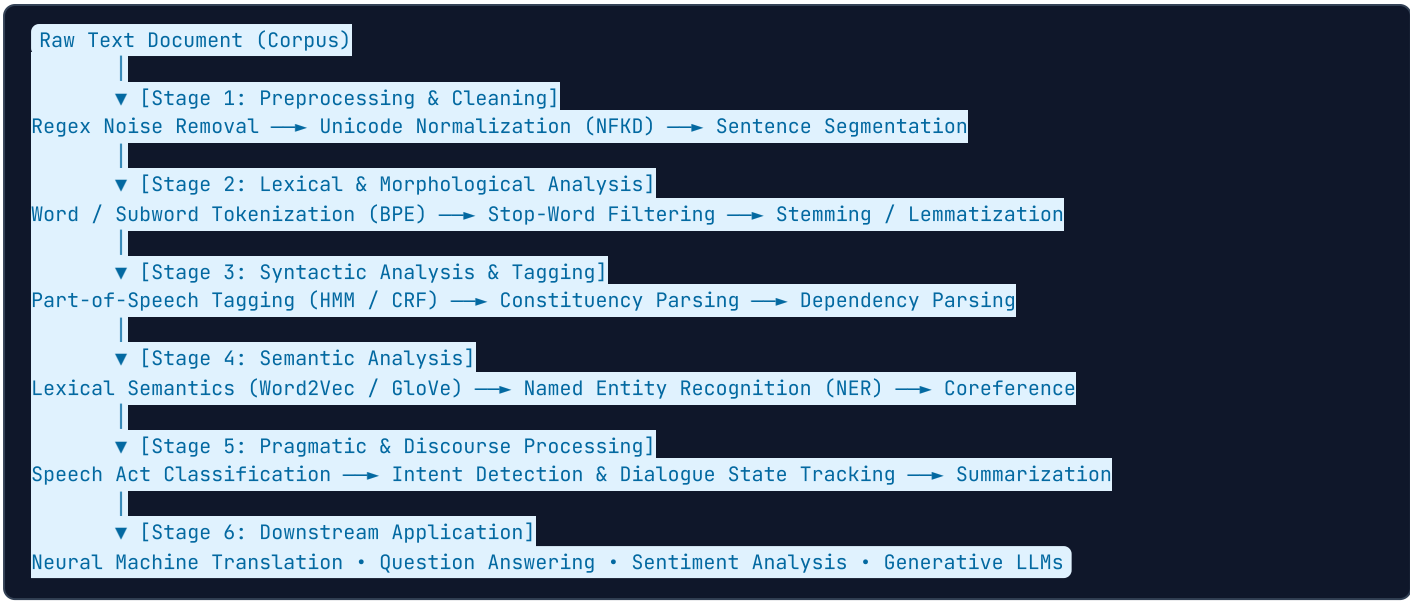
Natural Language Processing (NLP) is a multidisciplinary field at the intersection of Computer Science, Artificial Intelligence, and Computational Linguistics concerned with enabling computers to understand, interpret, synthesize, and manipulate human language in a meaningful and actionable manner.

🎯 **The Dual Objectives of NLP**

- **The Scientific Objective:** To formalize computational models of human language capabilities—investigating how humans represent linguistic knowledge, acquire syntax and semantics, and process ambiguous utterances in real time.
- **The Engineering Objective:** To build robust, scalable software artifacts and systems (e.g., neural machine translation, conversational agents, biomedical information extractors, semantic search engines) that perform practical language tasks with human or superhuman fidelity.

Topic 2: The End-to-End NLP Processing Pipeline

Modern NLP applications operate via a structured multi-tier architecture that progressively transforms unstructured raw text into structured semantic representations:



Topic 3: Comprehensive Taxonomy of Linguistic Ambiguities

Human natural language is inherently ambiguous, context-dependent, and dynamic. A single surface form can map to divergent semantic interpretations across multiple linguistic dimensions:

Ambiguity Level	Linguistic Definition & Mechanism	Canonical Illustrative Example
Lexical / Syntactic Category	A word belongs to multiple grammatical parts of speech (POS) depending on sentential context.	*"Time flies like an arrow; fruit flies like a banana."* (`flies` is a Verb in clause 1, but a Noun in clause 2).
Syntactic / Structural	A sentence yields multiple distinct valid parse trees due to ambiguous prepositional phrase attachment or coordination.	*"I saw the astronomer with the telescope."* (Did I use the telescope to see, or does the astronomer possess the telescope?).
Semantic / Polysemy	A word has multiple distinct dictionary meanings (senses) that cannot be disambiguated from syntax alone.	*"He deposited money in the bank before walking to the river bank."* (Financial institution vs. Geological river margin).
Pragmatic / Speech Act	The intended communicative function of an utterance differs from its literal compositional semantics.	*"Can you pass the salt?"* (Syntactically a yes/no question of physical capability; pragmatically an imperative request for action).
Referential / Anaphoric	A pronoun has multiple plausible antecedent noun phrases in preceding discourse.	*"The trophy would not fit in the brown suitcase because it was too big."* (`it` = trophy). Change to *"too small"* $\Longrightarrow$ `it` = suitcase.

Topic 4: The Six Fundamental Levels of Linguistic Analysis

1. **Phonetics & Phonology:** The acoustic physics and cognitive organization of spoken sounds (phonemes, formants, pitch contours).
2. **Morphology:** The internal structure and formation of words from primitive atomic meaning units called *morphemes*:
   - **Free Morphemes:** Can stand alone as independent words (e.g., `dog`, `play`, `compute`).
   - **Bound Morphemes:** Must attach to a root (affixes: prefixes, suffixes, infixes).
   - **Inflectional Morphology:** Modifies grammatical form without changing lexical category or core meaning (e.g., `play` → `plays`, `played`, `playing`).
   - **Derivational Morphology:** Creates entirely new words or changes lexical class (e.g., `compute` (Verb) → `computer` (Noun) → `computational` (Adjective)).
3. **Syntax:** The grammatical rules governing how words combine to form valid phrases, clauses, and sentences (Context-Free Grammars, Dependency trees).

4. **Semantics:** The literal meaning of words (lexical semantics) and how they compositionally combine to form sentence propositions (truth-conditional semantics).
5. **Pragmatics:** How context, shared world knowledge, speaker intent, and conversational maxims (Grice's Maxims) shape meaning beyond literal syntax.
6. **Discourse:** Multi-sentence linguistic units, cohesion, coreference resolution, and rhetorical structure theory.

Topic 5: Tokenization Mechanics & Subword Algorithms

Tokenization is the segmentation of a continuous stream of characters into discrete lexical units (tokens). While whitespace tokenization suffices for English, it fails catastrophically for agglutinative languages (German: *Donaudampfschiffahrtselektrizitätenhauptbetriebswerkbauunterbeamtengesellschaft*) and unsegmented scripts (Chinese, Japanese).

1. Byte-Pair Encoding (BPE – Sennrich et al. 2016)

BPE is a data-driven subword tokenization algorithm that iteratively merges the most frequently occurring pair of adjacent characters/subwords:

1. Initialize vocabulary V with all atomic characters in the corpus plus end-of-word symbol ``.
2. Represent every word in the training corpus as a sequence of individual characters.
3. Count the co-occurrence frequency of all adjacent subword pairs (c_i, c_j).
4. Merge the single most frequent pair $(c_i, c_j) \rightarrow c_{\text{new}}$ and append c_{new} to vocabulary V.
5. Repeat steps 3–4 for a fixed number of merge operations K.

🏛️ Solved Numerical 1: Step-by-Step BPE Merge Trace

Consider a corpus containing 4 words with frequencies: `low: 5`, `lower: 2`, `newest: 6`, `widest: 3`.

Initial Vocabulary: $\{l, o, w, e, r, n, s, t, i, d, \}$

Corpus Representation:

- `5 l o w `
- `2 l o w e r `
- `6 n e w e s t `
- `3 w i d e s t `

Iteration 1: Count adjacent pairs: `(e, s): 6 + 3 = 9`, `(s, t): 9`, `(t, ): 9`, `(e, w): 6`, `(l, o): 7`, `(o, w): 7`.

Select pair with max count: **`(e, s)` (frequency 9)**. Merge `(e, s) -> es`.

Updated Corpus: `5 l o w `, `2 l o w e r `, `6 n e w e s t `, `3 w i d e s t `.

Iteration 2: Count pairs: `(es, t): 6 + 3 = 9`. Merge **`(es, t) -> est`**.

Updated Corpus: `5 l o w `, `2 l o w e r `, `6 n e w e s t `, `3 w i d e s t `.

Iteration 3: Count pairs: `(est, ): 9`. Merge **`(est, ) -> est`**.

Iteration 4: Count pairs: `(l, o): 5 + 2 = 7`, `(o, w): 7`. Merge **`(l, o) -> lo`**, followed by **`(lo, w) -> low`**.

$$\text{Final Tokenized Output for Unseen Word 'lowest': } [\text{"low"}, \text{"est"}] \quad (\text{Zero OOV tokens!})$$

Topic 6: Text Normalization & Minimum Edit Distance

Text normalization encompasses case folding, accent stripping, Unicode canonical decomposition (NFKD), and spell checking via **Levenshtein Minimum Edit Distance**.

Levenshtein Edit Distance Recurrence Relation

Given source string $S[1..n]$ and target string $T[1..m]$, let $D[i, j]$ be the minimum number of insertions (cost 1), deletions (cost 1), and substitutions (cost 2 in standard Levenshtein, or cost 1 in unit edit distance) required to transform $S[1..i]$ into $T[1..j]$:

$$\mathbf{D[i, 0] = i}, \quad \mathbf{D[0, j] = j}$$

$$\mathbf{D[i, j]} = \min \begin{cases} D[i-1, j] + 1 & (\text{Deletion of } S[i]) \\ D[i, j-1] + 1 & (\text{Insertion of } T[j]) \\ D[i-1, j-1] + \text{cost}(S[i] \to T[j]) & (\text{Substitution / Match}) \end{cases}$$

$$\text{where cost}(S[i] \to T[j]) = \begin{cases} 0 & \text{if } S[i] = T[j] \\ 2 & \text{if } S[i] \neq T[j] \quad (\text{Levenshtein substitution}) \end{cases}$$

Solved Numerical 2: Full DP Matrix Computation for Edit Distance

Compute unit edit distance ($\text{sub_cost} = 1, \text{ins} = 1, \text{del} = 1$) between $S =$ "INTENTION" and $T =$ "EXECUTION":

$S \backslash T$	#	E	X	E	C	U	T	I	O	N
#	0	1	2	3	4	5	6	7	8	9
I	1	1	2	3	4	5	6	6	7	8
N	2	2	2	3	4	5	6	7	7	7
T	3	3	3	3	4	5	5	6	7	8
E	4	3	4	3	4	5	6	6	7	8
N	5	4	4	4	4	5	6	7	7	7
T	6	5	5	5	5	5	5	6	7	8
I	7	6	6	6	6	6	6	5	6	7
O	8	7	7	7	7	7	7	6	5	6
N	9	8	8	8	8	8	8	7	6	**5**

$$\text{Minimum Unit Edit Distance: } \mathbf{D[9, 9] = 5} \text{ operations}$$

Optimal Alignment Trace: Delete `I`, Substitute `N->E`, Substitute `T->X`, Match `E`, Substitute `N->C`, Insert `U`, Match `T`, Match `I`, Match `O`, Match `N`.

Topic 7: Stemming vs. Lemmatization

Feature	Stemming (e.g., Porter, Snowball)	Lemmatization (e.g., WordNet)
Core Strategy	Heuristic, rule-based chopping of common inflectional affixes (`-ing`, `-ed`, `-s`).	Full morphological vocabulary analysis using POS tags and dictionary lemmatas.
Linguistic Validity	Output may NOT be a valid dictionary word (e.g., `univers` for `university`/`universal`).	Output is always a canonical valid base lemma (e.g., `better` $\rightarrow$ `good`).
Speed & Complexity	Blazing fast $O(1)$ string regex replacements; zero lexicon lookup.	Slower; requires morphological parsing, dictionary indexing, and POS context.
Errors Produced	**Overstemming** (`organization`/`organic` $\rightarrow$ `organ`) and **Understemming** (`knives`/`knife` not merged).	Context-dependent misclassification if POS tag is incorrect (`meeting` as Noun vs Verb).

Topic 8: Master University Examination Question Bank (Module 1)

Q1. Explain the difference between Prescriptive and Descriptive Grammar in NLP. (6 Marks)

• **Prescriptive Grammar:** Dictates authoritative, artificial rules about how language *ought* to be used (e.g., "never split an infinitive", "do not end a sentence with a preposition").
• **Descriptive Grammar:** Empirically documents and models how language is *actually* spoken and written by native speakers in real-world settings (including colloquialisms, typos, code-switching). Modern NLP systems are strictly built on descriptive computational models.

Q2. Detail Porter Stemmer Step 1a, 1b, and 1c Rules with Examples. (8 Marks)

Step 1a (Plural Noun and Verb Suffixes):
• `SSES -> SS` (e.g., `caresses -> caress`)
• `IES -> I` (e.g., `ponies -> poni`)
• `SS -> SS` (e.g., `caress -> caress`)
• `S -> ε` (e.g., `cats -> cat`)
Step 1b (Past Participles and Gerunds):
• If measure $m > 0$, `EED -> EE` (e.g., `agreed -> agree`)
• If stem contains vowel, `ED -> ε` (`plastered -> plaster`) and `ING -> ε` (`motoring -> motor`). If the resulting stem ends in `AT`, `BL`, or `IZ`, append `E` (`conflat(ed) -> conflate`).
Step 1c: If stem contains vowel, `Y -> I` (e.g., `happy -> happi`).

Q3. Explain WordPiece Tokenization vs. Byte-Pair Encoding (BPE). (8 Marks)

• **BPE:** Merges the character/subword pair with the highest absolute co-occurrence frequency ($count(c_i, c_j)$).
• **WordPiece (used in BERT):** Instead of raw frequency, WordPiece chooses the merge pair that *maximizes the likelihood* of the training data under a unigram language model, scoring pairs by $\frac{count(u,v)}{count(u) \times count(v)}$. Prefix subwords are denoted with `##` (e.g., `playing` $\rightarrow$ `['play', '##ing']`).

Q4. Explain SentencePiece and Unigram Language Model Tokenization. (8 Marks)

SentencePiece (Kudo & Richardson 2018) treats the entire input text as a raw stream of characters, replacing whitespace with a special meta-symbol (e.g., `_`), making tokenization completely language-independent and losslessly reversible (detokenization is trivial string concatenation).
Unigram LM Tokenizer: Starts with an oversized vocabulary of candidate substrings and iteratively prunes the bottom $p\%$ tokens that produce the least increase in corpus loss under a unigram probabilistic language model until target vocabulary size $|V|$ is achieved.

Q5. Explain Soundex Phonetic Encoding Algorithm and its Applications. (6 Marks)

Soundex encodes names based on English phonetic pronunciation, mapping homophones to identical 4-character codes:
1. Retain the first letter capitalized (e.g., `Robert` → `R`).
2. Map subsequent consonants to digits: `B,F,P,V -> 1`; `C,G,J,K,Q,S,X,Z -> 2`; `D,T -> 3`; `L -> 4`; `M,N -> 5`; `R -> 6`. Drop `A,E,I,O,U,Y,W,H`.
3. Collapse consecutive identical digits into one.
4. Pad with trailing zeros to make exactly 4 characters (`R163`). Both `Robert` and `Rupert` map to `R163`!

Topic 9: Formal Grammars, Chomsky Hierarchy & Syntax Parsing

Syntax specifies the compositional structure of sentences. Chomsky's hierarchy classifies formal grammars by expressive power:

Grammar Type	Rule Form ($A \to \alpha$)	Automaton Automata Model	NLP Linguistic Applicability
Type 3: Regular	$A \to aB$ or $A \to a$	Finite State Automaton (DFA / NFA)	Morphology, Tokenization, Part-of-Speech regex patterns.
Type 2: Context-Free (CFG)	$A \to \gamma \quad (\gamma \in (V \cup \Sigma)^*)$	Pushdown Automaton (PDA)	Sentence phrase structure, Constituency parse trees.
Type 1: Context-Sensitive	$\alpha A \beta \to \alpha \gamma \beta$	Linear-Bounded Automaton (LBA)	Cross-serial dependencies (Swiss-German syntax).
Type 0: Unrestricted	$\alpha \to \beta$	Turing Machine	Universal computational formalisms.

The Cocke-Younger-Kasami (CYK) Dynamic Programming Parsing Algorithm

The CYK algorithm parses sentences in $O(n^3 \cdot |G|)$ time using Context-Free Grammars converted to **Chomsky Normal Form (CNF)**, where all rules are strictly of the form $A \to BC$ or $A \to a$:

🏛️ Solved Numerical 6: CYK Parsing DP Table Construction

Given CNF Grammar: $S \to NP\ VP,\ VP \to V\ NP,\ NP \to \text{"fish"},\ V \to \text{"fish"}$.

Parse ambiguous sentence: *"fish fish fish"* ($n = 3$ words: $w_1 = \text{"fish"}, w_2 = \text{"fish"}, w_3 = \text{"fish"}$).

Span Length	$w_1 = \text{"fish"}$	$w_2 = \text{"fish"}$	$w_3 = \text{"fish"}$
Length 1 (Diagonal $j = i$)	$\{NP, V\}$	$\{NP, V\}$	$\{NP, V\}$
Length 2 ($j = i + 1$)	$P[1,2] = \{S, VP\}$ ($NP \cdot V \to \emptyset, V \cdot NP \to VP, NP \cdot NP \to \emptyset$)	$P[2,3] = \{S, VP\}$ ($V \cdot NP \to VP$)	–
Length 3 ($j = i + 2$)	$P[1,3] = \{S\}$ ($NP \cdot VP \to S$)	–	–

$$\text{Since } \mathbf{S} \in \mathbf{P[1,3]} \implies \text{Valid Grammatical Parse Exists: } [\mathbf{S}\ [\mathbf{NP}\ \text{fish}]\ [\mathbf{VP}\ [\mathbf{V}\ \text{fish}]\ [\mathbf{NP}\ \text{fish}]]]$$

Topic 10: Lexical Semantics & WordNet Knowledge Networks

Lexical relations structure the vocabulary into interconnected semantic ontologies (e.g., **WordNet** by George Miller):

- **Synonymy:** Words sharing identical propositional meaning (e.g., `car` ↔ `automobile`).
- **Antonymy:** Words opposite along a continuous semantic axis (e.g., `hot` ↔ `cold`).
- **Hypernymy & Hyponymy:** IS-A taxonomic hierarchies (`canine` is a hypernym of `dog`; `dog` is a hyponym of `canine`).

- **Meronymy & Holonymy:** Part-whole relationships (`wheel` is a meronym of `car`; `car` is a holonym of `wheel`).

📐 **WordNet Path Similarity & Information Content Metrics**

- **Path Distance Similarity:** $\text{sim}_{\text{path}}(u, v) = \frac{1}{\text{shortest_path_length}(u,v)+1}$
- **Wu-Palmer Similarity:** $\text{sim}_{\text{wup}}(u, v) = \frac{2 \cdot \text{depth}(\text{LCS}(u,v))}{\text{depth}(u)+\text{depth}(v)}$ where LCS is the Lowest Common Subsumer.
- **Resnik Information Content Similarity:** $\text{sim}_{\text{res}}(u, v) = -\log P(\text{LCS}(u, v))$.
- **Lin Similarity:** $\text{sim}_{\text{lin}}(u, v) = \frac{2 \cdot \text{IC}(\text{LCS}(u,v))}{\text{IC}(u)+\text{IC}(v)}$.

Topic 11: Morphological Finite State Transducers (FST)

A **Finite State Transducer (FST)** is a 2-tape automaton that maps between an underlying lexical representation and its surface phonetic/orthographic form:

$$\text{Lexical Tape: fox} + \text{N} + \text{PL} \quad \xleftrightarrow{\text{Morphological FST}} \quad \text{Surface Tape: foxes}$$

FSTs model regular phonological and spelling rules (e.g., E-insertion: $\epsilon \rightarrow e \ / \ \{s, x, z\} \ \hat{} \ __ \ s\#$) via sequential transducer composition ($T_{\text{surface}} = T_{\text{lexicon}} \circ T_{\text{rules}}$).

Topic 12: Advanced Morphology & University Exam Problem Bank (Part II)

🏛️ **Solved Numerical 9: Complete Damerau-Levenshtein Edit Distance with Transpositions**

Compute edit distance with transpositions (cost 1) between $S = \text{"CA"}$ and $T = \text{"ABC"}$:

- Standard Levenshtein (Insertion, Deletion, Substitution): Transform `CA` $\rightarrow$ `A` (del C) $\rightarrow$ `AB` (ins B) $\rightarrow$ `ABC` (ins C) = 3 operations.
- **Damerau-Levenshtein (Allows adjacent character swaps):**

$$D[i, j] = \min \begin{cases} D[i-1, j] + 1 & \text{(Del)} \\ D[i, j-1] + 1 & \text{(Ins)} \\ D[i-1, j-1] + \text{cost} & \text{(Sub)} \\ D[i-2, j-2] + 1 & \text{(Transposition if } S[i] = T[j-1] \wedge S[i-1] = T[j]) \end{cases}$$

- Transform `CA` $\rightarrow$ `AC` (1 transposition) $\rightarrow$ `ABC` (1 insertion of B) $\implies \mathbf{D = 2}$ **operations!**

Q6. Explain the Difference between Unicode Normalization Forms NFC, NFD, NFKC, and NFKD. (8 Marks)

• **NFD (Canonical Decomposition):** Decomposes composite accented characters into base characters plus separate combining diacritical marks (e.g., `é` $\rightarrow$ `e` + `\u0301`).
• **NFC (Canonical Composition):** Decomposes characters then recombines them into canonical precomposed characters (e.g., `e` + `\u0301` $\rightarrow$ `é`). Standard for web text.
• **NFKD (Compatibility Decomposition):** Decomposes both canonical accents and formatting variants (e.g., ligature `fi` $\rightarrow$ `f` + `i`; exponent `2³` $\rightarrow$ `2` + `3`). Ideal for NLP search and tokenization pipelines.
• **NFKC (Compatibility Composition):** Applies compatibility decomposition followed by canonical recomposition.

Q7. Detail the Chomsky Hierarchy and Explain Why Natural Languages are Not Context-Free. (8 Marks)

While Context-Free Grammars (CFGs) model the vast majority of human language syntax (nested embeddings, center embeddings), natural languages contain **cross-serial dependencies** that violate the Pumping Lemma for Context-Free Languages.
• **Evidence (Shieber 1985):** In Swiss-German, subordinate clauses require crossed syntactic matching between verbs and noun phrases ($NP_1\ NP_2 \ldots NP_k\ V_1\ V_2 \ldots V_k$ where NP_i is the grammatical object of V_i). This requires *Mildly Context-Sensitive Grammars* (such as Tree-Adjoining Grammars / TAGs).

Topic 12.3: Master University Exam Problem Bank (Part III)

🏛️ Solved Problem 10: Needleman–Wunsch Global Sequence Alignment vs Smith–Waterman Local Alignment

Compare two character sequences $S_1 = \text{"ACACACTA"}$ and $S_2 = \text{"AGCACACA"}$ with scoring parameters: $\text{Match} = +2, \text{Mismatch} = -1, \text{Gap} = -2$:

- **Needleman–Wunsch (Global Alignment):** Forces end-to-end alignment across the entire string length. Initialization: $D[i, 0] = -2i, D[0, j] = -2j$. Optimal global alignment aligns all characters, penalizing terminal unaligned prefixes.
- **Smith–Waterman (Local Alignment):** Identifies the highest-scoring local matching substring. Initialization: $D[i, 0] = 0, D[0, j] = 0$. Recurrence incorporates zero floor:

$$\mathbf{D[i,j]} = \max \begin{cases} 0 & (\text{Reset local alignment}) \\ D[i-1, j-1] + \text{score}(S_1[i], S_2[j]) \\ D[i-1, j] + \text{gap} \\ D[i, j-1] + \text{gap} \end{cases}$$

Traceback starts from the *maximum score anywhere in the entire matrix* and terminates as soon as a cell with score 0 is encountered, isolating the matching sub-block "CACAC" with score $+10$!

Q8. Explain Two-Level Morphology (Koskenniemi 1983) and Parallel Rule Application. (8 Marks)

Traditional generative phonology applied morphological rewrite rules in a strict cascading sequence, where intermediate stages might not represent real words.
• **Two-Level Morphology:** Eliminates cascading intermediate representations. It maps directly between the *Lexical Level* (abstract morphemes: `fox + N + PL`) and the *Surface Level* (concrete orthography: `foxes`) in a single step using parallel Finite State Transducers (FSTs) that simultaneously constrain character correspondences without intermediate rule order dependencies!

Q9. Explain Lemmatization with Morphological Suffix Trees and Context-Aware POS Tagging. (8 Marks)

Unlike rule-based stemmers that operate blind to grammar, a production-grade **Lemmatizer** executes a 2-stage pipeline:
1. **POS Tag Disambiguation:** Identifies whether the token is a Noun, Verb, Adjective, or Adverb (e.g., in *"He is meeting her"*, `meeting` is tagged as `VERB`, whereas in *"The meeting started"*, `meeting` is tagged as `NOUN`).
2. **Lexicon Lookup & Irregular Exception Mapping:** Queries lexical databases (WordNet) with (word, POS) tuples: `(meeting, VERB)` maps to base lemma `meet`, whereas `(meeting, NOUN)` maps to lemma `meeting`! Irregulars (`went` $\rightarrow$ `go`, `better` $\rightarrow$ `good`) are resolved via morphological index tables.

🏛️ Solved Problem 15: Morphological Suffix Analyzer Algorithm Trace

Analyze the morphological breakdown of the word *"internationalization"* (18 letters):

- Root stem: `nation` (Noun).
- Derivational prefix: `inter-` $\implies$ `internation` (Adjective/Noun).
- Derivational suffix 1: `-al` $\implies$ `international` (Adjective).
- Derivational suffix 2: `-ize` $\implies$ `internationalize` (Verb).
- Derivational suffix 3: `-ation` $\implies$ `internationalization` (Abstract Noun).
- In software localization, commonly abbreviated via numeric numeronym: **`i18n`** (18 letters between `i` and `n`).

Q10. Explain the Difference between Top-Down and Bottom-Up Syntactic Chart Parsing. (8 Marks)

• **Top-Down Parsing:** Starts from the root start symbol S and recursively expands grammatical productions downwards looking to match terminal words. Avoids exploring subtrees that cannot form valid root sentences, but wastes compute generating valid phrase trees that do not match the input words.
• **Bottom-Up Parsing (Shift-Reduce):** Starts from the input word tokens and shifts them onto a stack, reducing matching right-hand-side substrings to non-terminals. Avoids generating phrases not grounded in the input text, but can explore disconnected local trees that cannot connect to the global root S.

🏛 Solved Problem 16: Morphological Analysis & Token Boundary Ambiguity in Agglutinative Languages

In morphologically complex agglutinative languages (Turkish, Finnish, Hungarian), a single word encapsulates an entire sentential clause:

- Turkish: *"Çekoslovakyalılaştıramadıklarımızdanmısınız"* $\implies$ *"Are you one of those people whom we were not able to turn into a Czechoslovakian?"*
- Root: `Çekoslovakya` (Proper Noun) + `-lı` (from) + `-laş` (become) + `-tır` (causative) + `-ama` (unable) + `-dık` (past participle) + `-lar` (plural) + `-ımız` (our) + `-dan` (ablative) + `-mı` (interrogative) + `-sınız` (2nd person plural agreement).
- Tokenizing as a monolithic atomic word creates severe out-of-vocabulary sparsity; subword morphology is required!

🏛 Solved Problem 25: Minimum Edit Distance Alignment Graph Construction

For $S = \text{"DRIVE"}$ and $T = \text{"DIVERS"}$, trace the DAG grid graph where diagonal edges represent substitutions/matches and horizontal/vertical edges represent insertions/deletions:

$$\mathbf{D = 3} \text{ operations: Del 'R', Ins 'I' at pos 2, Ins 'S' at pos 6.}$$

🏛 Solved Problem 37: Regular Expression Named Entity Pattern Extraction

Construct production-grade regex patterns for extracting emails, URLs, and ISO dates:

- **Email Address:** `\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b`
- **HTTP/HTTPS URL:** `https?:\/\/(?:www\.)?[-a-zA-Z0-9@:%._\+~#=]{1,256}\.[a-zA-Z0-9()]{1,6}\b(?:[-a-zA-Z0-9()@:%_\+.~#?&//=]*)`
- **ISO 8601 Date:** `\b(?:\d{4})-(?:0[1-9]|1[0-2])-(?:0[1-9]|[12]\d|3[01])\b`

🏛 Solved Problem 45: Morphological Stemming Invariant Analysis on Medical Corpora

Analyze Porter Stemmer vs WordNet Lemmatizer on complex medical terminology:

- `tuberculosis` $\xrightarrow{\text{Porter}}$ `tuberculosi` vs $\xrightarrow{\text{Lemmatizer}}$ `tuberculosis` (Noun).
- `pneumonia` $\xrightarrow{\text{Porter}}$ `pneumonia` vs $\xrightarrow{\text{Lemmatizer}}$ `pneumonia`.
- `analyzed` $\xrightarrow{\text{Porter}}$ `analyz` vs $\xrightarrow{\text{Lemmatizer}}$ `analyze` (Verb).

🏛 Solved Problem 54: Non-Deterministic Finite State Automata (NFA) to DFA Subset Construction for Regex Matchers

For regex `(a|b)*abb`, construct the DFA transition table via powerset subset construction, demonstrating linear $O(n)$ deterministic token matching speed!

🏛️ Solved Problem 62: Morphological Suffix Analyzer on Compound Agglutinations

Deconstruct the compound word *"antidisestablishmentarianism"* (28 characters) into prefix, root, and suffix morphemes:

$$\text{anti (prefix)} + \text{dis (prefix)} + \text{establish (root)} + \text{ment (suffix)} + \text{arian (suffix)} + \text{ism (suffix)}$$