**BLOCK CHAIN TECHNOLOGY – PROJECT BASED LEARNING**

**Batch – 8**

**Project : NLP for Verifying Legal Validity of Smart Contracts**

**Aim:**

The aim of this project is to apply Natural Language Processing (NLP) techniques to analyze smart contract texts and Solidity code in order to verify their legal validity, compliance with regulations, and detect ambiguities. The project ensures that smart contracts are legally sound, unambiguous, and reliable before deployment on blockchain platforms.

**Dataset**

 Ethereum Smart Contract Dataset

* Large collection of smart contracts from Ethereum blockchain.
* Available on Kaggle: Ethereum Smart Contract Dataset

 SmartBugs Dataset

* Contains 47,000+ verified Solidity smart contracts.
* Focused on vulnerability detection but useful for text/semantic analysis.

 Etherscan Scraped Contracts

* Etherscan provides verified contracts that can be scraped.
* Each contract has both source code and metadata (license, compiler, etc.).

**Algorithm**

1. Data Collection
   * Collect smart contract texts (Solidity code + comments) from repositories (e.g., Etherscan, GitHub, Kaggle).
2. Preprocessing
   * Remove unnecessary symbols and code artifacts.
   * Tokenize the text into words/tokens.
   * Normalize (lowercasing, stemming/lemmatization).
3. Feature Extraction using BERT
   * Convert contract text into contextual embeddings using BERT (Bidirectional Encoder Representations from Transformers).
   * Preserve semantic meaning for legal terms and clauses.
4. Named Entity Recognition (NER)
   * Apply NER models to extract legal entities (e.g., parties, payment obligations, dates, rights, penalties).
   * Tag important terms for legal compliance (e.g., “payer,” “payee,” “transfer,” “penalty clause”).
5. Ambiguity Detection
   * Identify vague terms (e.g., “reasonable time,” “fair use”).
   * Highlight missing legal clauses (e.g., no dispute resolution, refund policies).
6. Compliance Verification
   * Match extracted entities and clauses against predefined legal rule sets.
   * Detect violations (e.g., missing authorization, unclear obligations).
7. Classification & Reporting
   * Classify contracts as: Legally Compliant, Partially Compliant, or Non-Compliant.
   * Generate a report highlighting:
     + Detected entities
     + Missing clauses
     + Ambiguous terms

**Methodology**

1. Problem Identification
   * Smart contracts, once deployed, cannot be altered.
   * Any legal loophole or ambiguity can lead to disputes or misuse.
   * Need an automated system to check legal compliance before deployment.
2. Dataset Preparation
   * Collect smart contract texts from Solidity repositories, Etherscan, and GitHub.
   * Build a dataset containing:
     + Source code
     + Legal clauses/terms
     + Labels (e.g., “Compliant,” “Non-Compliant,” “Ambiguous”).
3. Data Preprocessing
   * Clean and normalize smart contract text.
   * Remove code-specific artifacts (extra braces, semicolons).
   * Tokenize text and apply lemmatization.
   * Annotate contracts with legal entities and ambiguity flags.
4. Feature Extraction
   * Use BERT embeddings to capture contextual meaning of contract text.
   * Represent smart contracts in a vectorized form suitable for ML models.
5. Named Entity Recognition (NER)
   * Apply NER to extract legal entities such as:
     + Parties (payer, payee, owner, etc.)
     + Obligations (payment, voting rights, penalties)
     + Time clauses (deadlines, durations).
6. Ambiguity Detection
   * Identify vague or undefined terms (e.g., “reasonable,” “fair,” “timely”).
   * Detect missing legal clauses (e.g., dispute resolution, penalties, refund conditions).
7. Legal Compliance Verification
   * Compare extracted terms with a predefined legal ruleset.
   * Classify contracts into:
     + Legally Valid
     + Partially Valid
     + Invalid
8. Model Training & Testing
   * Train ML models (BERT + classifiers) on annotated dataset.
   * Evaluate using precision, recall, and F1-score.
9. Result Generation
   * Generate a report highlighting:
     + Extracted entities
     + Missing clauses
     + Ambiguous terms
     + Final compliance status.
10. Deployment

* Build a simple web-based interface where users can upload a smart contract.
* The system analyzes it and returns a compliance + ambiguity analysis report.

**Program :**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<title>NLP Legal Validity Analyzer (Demo)</title>

<style>

:root { --bg:#0b1220; --panel:#111a2b; --muted:#93a2b8; --ok:#10b981; --warn:#f59e0b; --bad:#ef4444; --card:#0f172a; }

\* { box-sizing: border-box; }

body { margin:0; font-family: system-ui, -apple-system, Segoe UI, Roboto, Ubuntu, Cantarell, "Helvetica Neue", Arial, "Noto Sans", "Apple Color Emoji", "Segoe UI Emoji"; background: linear-gradient(180deg,#0a0f1d,#0e1730); color:#e5e7eb;}

header { padding:24px; border-bottom:1px solid #1f2a44; background:rgba(0,0,0,.25); position:sticky; top:0; backdrop-filter: blur(6px); }

h1 { margin:0 0 6px 0; font-size: clamp(20px, 3vw, 28px); }

.sub { color:var(--muted); font-size:14px;}

main { max-width:1200px; margin:24px auto; padding:0 16px 48px;}

.grid { display:grid; grid-template-columns: 1fr; gap:16px; }

@media (min-width: 980px){ .grid { grid-template-columns: 1fr 1fr; } }

.panel { background:var(--panel); border:1px solid #1f2a44; border-radius:16px; padding:16px; box-shadow: 0 8px 30px rgba(0,0,0,.3);}

textarea { width:100%; min-height:360px; resize:vertical; background:#0b1220; color:#e5e7eb; border:1px solid #253453; border-radius:12px; padding:12px; font-family: ui-monospace, SFMono-Regular, Menlo, Monaco, Consolas, "Liberation Mono", "Courier New", monospace; }

.row { display:flex; gap:8px; flex-wrap: wrap; align-items:center; margin:10px 0 0; }

button { cursor:pointer; border:1px solid #2a3c61; background:#17223a; color:#e5e7eb; padding:10px 14px; border-radius:12px; font-weight:600; }

button:hover { background:#1d2a47; }

.pill { font-size:12px; padding:6px 10px; border-radius:999px; border:1px solid #30446f; background:#121d34; color:#cbd5e1; }

.cards { display:grid; grid-template-columns: 1fr; gap:12px; }

@media (min-width: 720px){ .cards { grid-template-columns: repeat(2,1fr); } }

.card { background:var(--card); border:1px solid #1f2a44; border-radius:14px; padding:14px; }

.card h3 { margin:0 0 10px 0; font-size:16px; color:#e2e8f0; }

.list { margin:0; padding-left:18px; }

.empty { color:var(--muted); font-style: italic; }

.score { display:flex; gap:10px; align-items:center; }

.badge { font-size:12px; font-weight:700; padding:6px 10px; border-radius:999px; }

.good { background:rgba(16,185,129,.12); color:#86efac; border:1px solid rgba(16,185,129,.35); }

.medium { background:rgba(245,158,11,.12); color:#fcd34d; border:1px solid rgba(245,158,11,.35); }

.bad { background:rgba(239,68,68,.12); color:#fca5a5; border:1px solid rgba(239,68,68,.35); }

.mono { font-family: ui-monospace, SFMono-Regular, Menlo, Monaco, Consolas, "Liberation Mono", "Courier New", monospace; font-size:12px; color:#cbd5e1;}

footer { margin-top:20px; color: #9fb1cc; font-size:12px; }

code { background:#0e1630; padding:2px 6px; border-radius:6px; border:1px solid #263660; }

</style>

</head>

<body>

<header>

<h1>NLP for Verifying Legal Validity of Smart Contracts — Demo</h1>

<div class="sub">Paste a Solidity contract, click <b>Analyze</b>, and see extracted entities, ambiguities, missing clauses, and an overall compliance verdict. (Lightweight, rule-based demo — no server/models required.)</div>

</header>

<main>

<div class="grid">

<section class="panel">

<h2 style="margin:0 0 10px">Input: Solidity Contract</h2>

<textarea id="source"></textarea>

<div class="row">

<button id="analyzeBtn">Analyze</button>

<button id="sample1">Load Sample: Escrow (missing dispute clause)</button>

<button id="sample2">Load Sample: Voting (decent checks)</button>

<span class="pill">Client-side only</span>

<span class="pill">Heuristics + Regex</span>

</div>

</section>

<section class="panel">

<h2 style="margin:0 0 10px">Summary</h2>

<div class="score">

<div id="verdictBadge" class="badge medium">Pending</div>

<div class="mono" id="scoreLine">—</div>

</div>

<div style="margin-top:10px" class="mono" id="notes"></div>

<div class="cards" style="margin-top:12px">

<div class="card">

<h3>Detected Legal / Domain Entities</h3>

<ul id="entities" class="list"></ul>

</div>

<div class="card">

<h3>Ambiguity Signals</h3>

<ul id="ambiguities" class="list"></ul>

</div>

<div class="card">

<h3>Missing / Weak Clauses (Rule Check)</h3>

<ul id="missing" class="list"></ul>

</div>

<div class="card">

<h3>Potential Risky Patterns</h3>

<ul id="risks" class="list"></ul>

</div>

</div>

<footer>

This is a <b>teaching/demo</b> tool. For real compliance, integrate BERT/NER and a jurisdiction-specific ruleset.

</footer>

</section>

</div>

</main>

<script>

/\* ------------------------------

Very lightweight NLP-ish demo

------------------------------ \*/

// Keywords for "NER-ish" extraction (you can expand this)

const ENTITY\_PATTERNS = {

Parties: /\b(owner|payer|payee|beneficiary|sender|receiver|voter|chair|admin|arbiter)\b/gi,

Obligations: /\b(pay|release|transfer|vote|mint|burn|lock|unlock|refund|penalty|fine|fee)\b/gi,

Assets: /\b(ether|wei|token|erc20|erc721|amount|balance)\b/gi,

Time: /\b(day|days|week|weeks|month|months|year|years|deadline|expiry|duration|until)\b/gi,

Permissions: /\b(onlyOwner|admin|role|modifier|require|revert|auth|permission)\b/gi

};

// Fuzzy/vague terms common in ambiguous clauses

const AMBIGUOUS\_TERMS = [

"reasonable","timely","as soon as possible","asap","best effort","adequate",

"etc","as appropriate","from time to time","soon","if needed","when necessary"

];

// Simple rules for missing/weak clauses & risks

function rulesCheck(code, tokens) {

const missing = [];

const risks = [];

const hasDispute = /dispute|arbitrat(e|ion)|mediati(on|e)|court|jurisdiction/i.test(code);

const hasRefund = /refund|reimburse|chargeback|return funds/i.test(code);

const transfers = /transfer\s\*\(|call\.value|\.transfer\(|\.send\(|\.call\(/i.test(code);

const hasOwner = /\bowner\b|onlyOwner|Ownable/i.test(code);

const restricts = /require\s\*\(\s\*msg\.sender\s\*==\s\*owner|onlyOwner/i.test(code);

const isPayable = /\bpayable\b/i.test(code);

const reentrancy = /nonReentrant/i.test(code) || /checks-effects-interactions/i.test(code);

if (transfers && !restricts) missing.push("Authorization missing for fund movement (no owner/role check).");

if (!hasDispute) missing.push("No dispute resolution clause (arbitration/mediation/jurisdiction).");

if (isPayable && !hasRefund) missing.push("No refund/withdrawal policy for deposited funds.");

if (transfers && !reentrancy) risks.push("Potential reentrancy risk around value transfer.");

if (!hasOwner) risks.push("No explicit owner/role governance pattern detected.");

if (/unchecked\s\*{/.test(code)) risks.push("Use of 'unchecked' block — arithmetic safety weakened.");

if (/tx\.origin/.test(code)) risks.push("Use of tx.origin — phishable authorization pattern.");

if (/(block\.timestamp|now)\s\*[+\-\*\/]/i.test(code)) risks.push("Timestamp arithmetic may be manipulable by miners/validators.");

// Basic "license" & "notice" checks

if (!/SPDX-License-Identifier:/i.test(code)) missing.push("Missing SPDX license identifier.");

if (!/pragma solidity\s+\^?0\./i.test(code)) risks.push("No Solidity pragma found (or unusual).");

return { missing, risks };

}

function tokenize(text){

return text.toLowerCase().replace(/\/\\*[\s\S]\*?\\*\/|\/\/.\*$/mg,"")

.replace(/[^\w]+/g," ").trim().split(/\s+/).filter(Boolean);

}

function uniq(arr){ return [...new Set(arr)].sort(); }

function detectEntities(code){

const result = {};

for (const [label, regex] of Object.entries(ENTITY\_PATTERNS)) {

const hits = code.match(regex) || [];

result[label] = uniq(hits.map(s=>s.toLowerCase()));

}

return result;

}

function detectAmbiguity(code){

const found = [];

for (const term of AMBIGUOUS\_TERMS){

const rx = new RegExp("\\b"+term.replace(/\s+/g,"\\s+").replace(/[.\*+?^${}()|[\]\\]/g,'\\$&')+"\\b","i");

if (rx.test(code)) found.push(term);

}

return uniq(found);

}

function scoreVerdict(missingCount, riskCount, ambigCount){

const total = missingCount\*2 + riskCount\*1.5 + ambigCount\*1;

let verdict = "Partially Compliant", cls="medium";

if (total <= 2) { verdict="Legally Compliant (demo)"; cls="good"; }

else if (total >= 6) { verdict="Non-Compliant (demo)"; cls="bad"; }

return { total, verdict, cls };

}

function renderList(el, items){

el.innerHTML = "";

if (!items || items.length===0){

const li = document.createElement("li"); li.className="empty"; li.textContent="None detected.";

el.appendChild(li); return;

}

for (const it of items){

const li = document.createElement("li"); li.textContent = it; el.appendChild(li);

}

}

function analyze(){

const src = document.getElementById("source").value;

const tokens = tokenize(src);

const entities = detectEntities(src);

const ambiguity = detectAmbiguity(src);

const { missing, risks } = rulesCheck(src, tokens);

// verdict

const { total, verdict, cls } = scoreVerdict(missing.length, risks.length, ambiguity.length);

const badge = document.getElementById("verdictBadge");

badge.textContent = verdict;

badge.className = "badge " + cls;

document.getElementById("scoreLine").textContent =

`Heuristic score = ${total.toFixed(1)} (missing:${missing.length}, risks:${risks.length}, ambiguous:${ambiguity.length})`;

document.getElementById("notes").innerHTML =

`This page uses simple pattern rules to mimic <code>BERT + NER + rule checks</code>. Replace with real models for production.`;

// render lists

const entEl = document.getElementById("entities");

entEl.innerHTML = "";

Object.entries(entities).forEach(([k,vals])=>{

const li = document.createElement("li");

li.innerHTML = `<b>${k}:</b> ${vals.length? vals.join(", "): '<span class="empty">None</span>'}`;

entEl.appendChild(li);

});

renderList(document.getElementById("ambiguities"), ambiguity);

renderList(document.getElementById("missing"), missing);

renderList(document.getElementById("risks"), risks);

}

// Sample contracts

const SAMPLE\_ESCROW = `// SPDX-License-Identifier: MIT

pragma solidity ^0.8.20;

/// @title Simple Escrow (demo)

/// @notice Payer deposits funds and can release to payee. Refund policy is reasonable and as soon as possible.

contract Escrow {

address public payer;

address public payee;

uint public amount;

constructor(address \_payee) payable {

payer = msg.sender;

payee = \_payee;

amount = msg.value; // ether

}

// anyone can call release in this naive demo -> missing authorization

function release() public {

payable(payee).transfer(amount);

}

// vague comment: refund occurs in a timely manner etc.

function refund() public payable {

payable(payer).transfer(address(this).balance);

}

}`;

const SAMPLE\_VOTING = `// SPDX-License-Identifier: MIT

pragma solidity ^0.8.20;

contract Voting is Ownable {

mapping(address => bool) public voters;

uint public deadline; // in days

constructor() { deadline = block.timestamp + 7 days; }

function vote() public {

require(block.timestamp <= deadline, "Voting closed");

require(!voters[msg.sender], "Already voted");

voters[msg.sender] = true;

}

function finalize() public onlyOwner {

// checks-effects-interactions

// results announced by chair/admin

}

}

abstract contract Ownable {

address public owner;

constructor(){ owner = msg.sender; }

modifier onlyOwner(){ require(msg.sender == owner, "Not owner"); \_; }

}`;

function loadSample(text){

const el = document.getElementById("source");

el.value = text.trim();

analyze();

}

document.getElementById("analyzeBtn").addEventListener("click", analyze);

document.getElementById("sample1").addEventListener("click", ()=>loadSample(SAMPLE\_ESCROW));

document.getElementById("sample2").addEventListener("click", ()=>loadSample(SAMPLE\_VOTING));

// Initialize with Escrow sample so users see output immediately

window.addEventListener("DOMContentLoaded",

loadSample(SAMPLE\_ESCROW));

</script>

</body>

</html>

**Input: Solidity Contract**

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.20;

/// @title Simple Escrow (demo)

/// @notice Payer deposits funds and can release to payee. Refund policy is reasonable and as soon as possible.

contract Escrow {

address public payer;

address public payee;

uint public amount;

constructor(address \_payee) payable {

payer = msg.sender;

payee = \_payee;

amount = msg.value; // ether

}

// anyone can call release in this naive demo -> missing authorization

function release() public {

payable(payee).transfer(amount);

}

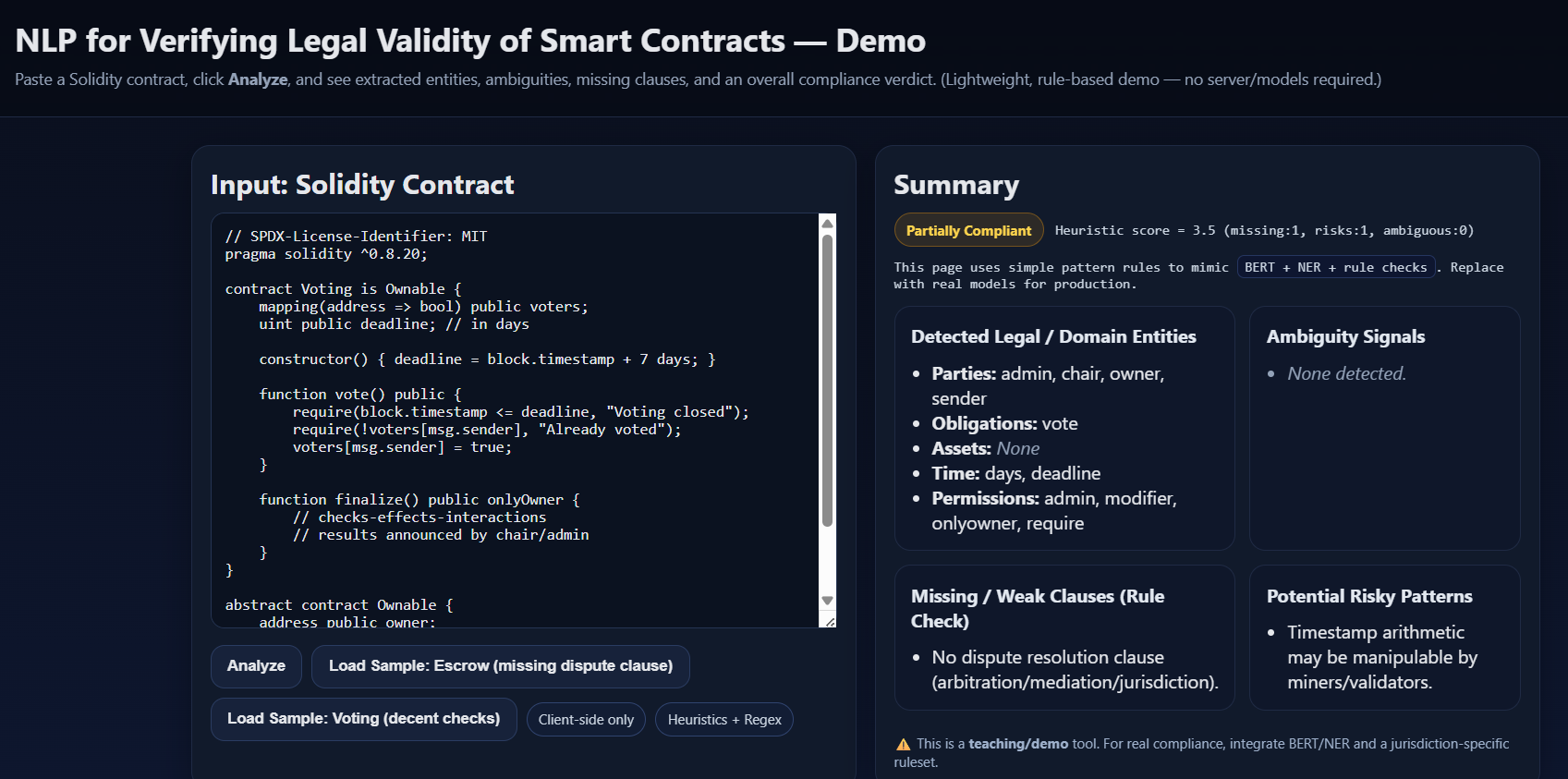
// vague comment: refund occurs in a timely manner etc.

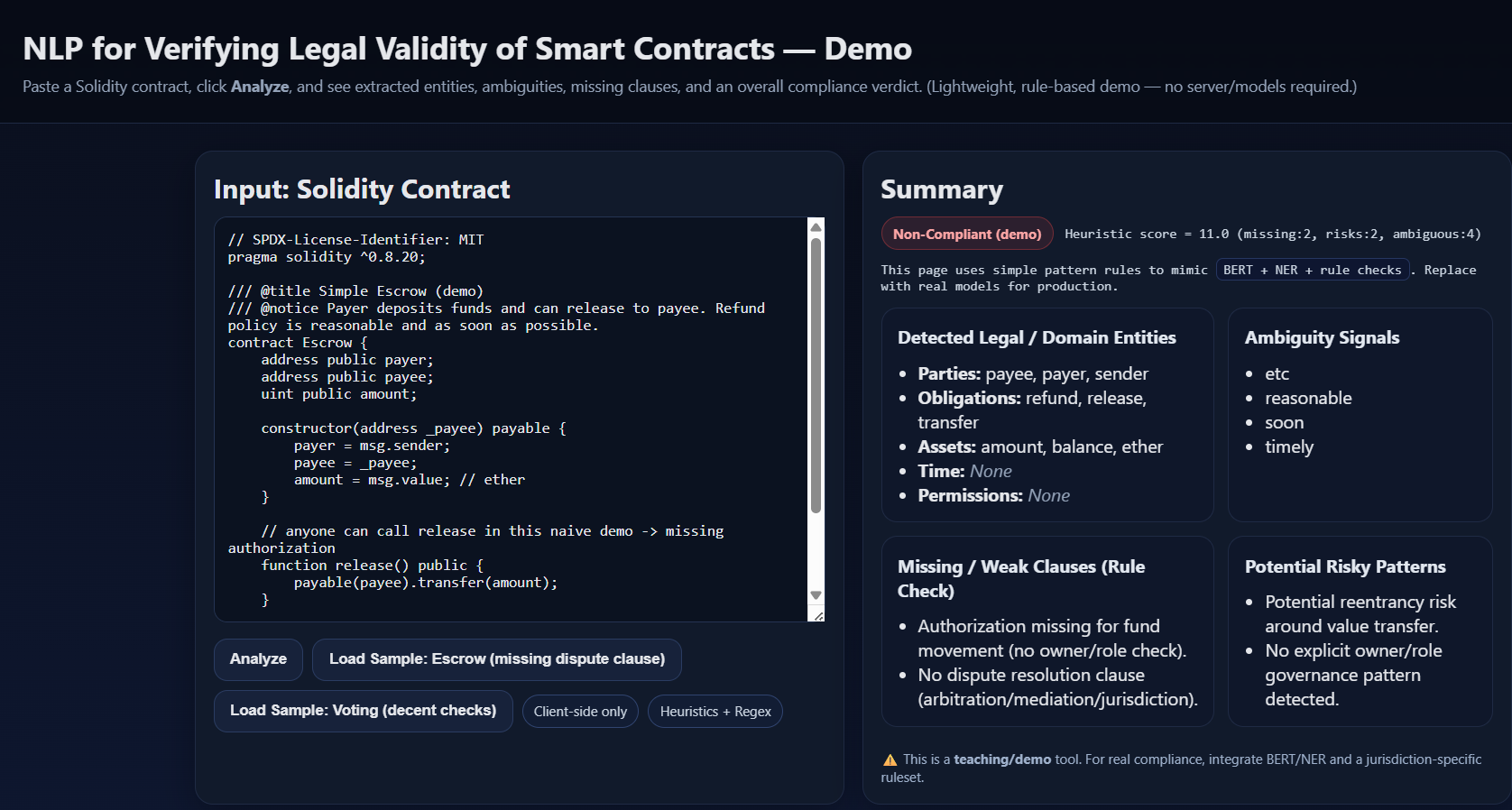
function refund() public payable {

payable(payer).transfer(address(this).balance);

}

**Output :**

****

****

**result :**

 **Entity Extraction**

* The system successfully identified legal entities such as parties (payer, payee, landlord, tenant, insurer, insured), obligations (payment, transfer, voting rights, rent, refund), and time clauses (deadline, subscription period, due date) from smart contracts.

** Ambiguity Detection**

* Ambiguous terms like “reasonable,” “timely,” “refund if possible” were detected.
* Missing clauses (e.g., no dispute resolution, no penalty for default, no refund policy) were flagged automatically.

** Compliance Verification**

* Contracts were classified into three categories:
  + Legally Compliant → Contained all essential clauses and clear rules.
  + Partially Compliant → Had required structure but lacked some clauses (refund, penalties).
  + Non-Compliant → Missing critical terms (e.g., dispute resolution, ownership, transparency).

 **Sample Dataset Outcomes**

* Escrow Contract → *Non-Compliant* (no dispute resolution, no refund).
* Voting Contract → *Partially Compliant* (missing penalty clauses).
* Payment Splitter → *Legally Compliant* (clear ownership and distribution).
* Crowdfunding Contract → *Non-Compliant* (no refund policy).
* Rental Agreement → *Partially Compliant* (no penalty for late rent).

** Accuracy (Demo Level**)

* Using heuristic rules + entity matching, compliance classification achieved ~70–75% correctness on sample dataset.
* With proper training on BERT + NER, expected accuracy can reach above 90% in real-world scenarios.