i like this table so simple ... 🎯 Summary: What Each Extraction Specializes In Extraction Method Specializes In Simple Test Pipeline Impact Multi-level Segmentation Breaking complex text into claims "Multiple actions in one sentence?" Prevents missing critical claims Context-aware Boundaries Finding where to split/merge "But/however/then in text?" Ensures proper claim isolation Discourse Analysis Understanding relationships "Because/therefore in text?" Catches misleading justifications Hierarchical Typing Categorizing claim types "Security/command/reference?" Routes to right verification engine Semantic Understanding Deep command parsing "Technical command syntax?" Catches incomplete/partial commands Claim Context Isolation Separating facts from references "Cites standards/docs?" Prevents citation misrepresentations Logical Flow Analysis Analyzing conditions/dependencies "If/then logic?" Prevents runtime logic failures

**🎯 Complete 7-Extraction Specialization Table**

| Extraction Method | Specializes In | Simple Test | Pipeline Impact | Real Example |
| --- | --- | --- | --- | --- |
| **Multi-level Segmentation** | Breaking complex text into individual claims | "Multiple actions in one sentence?" | Prevents missing critical security claims | "aws s3 encrypt & privileged true" → Separate commands |
| **Context-aware Boundaries** | Finding where to split or merge claims | "But/however/then in text?" | Ensures proper claim isolation and context | "yum but dnf on AL2023" → Split at contrast |
| **Discourse Analysis** | Understanding rhetorical relationships | "Because/therefore in text?" | Catches misleading justifications and reasoning | "MFA because CIS requires" → Map justification |
| **Hierarchical Typing** | Categorizing claims by type and risk | "Security/command/reference?" | Routes to correct verification engine | "64Mi Java" → JVM memory config |
| **Semantic Understanding** | Deep parsing of technical command syntax | "Technical command syntax?" | Catches incomplete/partial commands | "trivy severity" → Missing ignore-unfixed |
| **Claim Context Isolation** | Separating facts from references/citations | "Cites standards/docs?" | Prevents citation misrepresentations | "CIS requires" → Separate fact from citation |
| **Logical Flow Analysis** | Analyzing conditions and dependencies | "If/then logic?" | Prevents runtime logic failures | "if $CI\_TAG" → Unreliable variable |

**🎯 Extraction → Identification → Resolution Flow Table**

| Extraction Method | Extraction Output | Identification Result | Resolution Action |
| --- | --- | --- | --- |
| **Multi-level Segmentation** | Separate individual claims | Verify each claim independently | Fix each invalid claim specifically |
| **Context-aware Boundaries** | Claims with proper context | Check context-appropriate validity | Apply context-aware corrections |
| **Discourse Analysis** | Mapped relationships | Validate justification accuracy | Correct misleading relationships |
| **Hierarchical Typing** | Categorized claim types | Type-specific verification | Type-appropriate fixes |
| **Semantic Understanding** | Deep command analysis | Syntax + completeness check | Complete partial commands |
| **Claim Context Isolation** | Separated facts & citations | Verify citation accuracy | Correct attribution errors |
| **Logical Flow Analysis** | Logic reliability assessment | Check variable/condition reliability | Implement robust logic patterns |

**🎯 Quick Decision Matrix: Which Extraction When?**

| If the claim contains... | Use This Extraction | Because it specializes in... |
| --- | --- | --- |
| **Multiple actions/commands** | Multi-level Segmentation | Breaking complex text into individual verifiable claims |
| **Contrast/sequence markers** | Context-aware Boundaries | Understanding where claims should split or merge |
| **Justification/reasoning** | Discourse Analysis | Mapping rhetorical relationships and support |
| **Technical configurations** | Hierarchical Typing | Categorizing by risk level and verification needs |
| **Command syntax/tools** | Semantic Understanding | Deep parsing of technical command structure |
| **Standards/citations** | Claim Context Isolation | Separating factual claims from source references |
| **Conditional logic** | Logical Flow Analysis | Analyzing dependencies and reliability |

**This table shows how each extraction method is a specialist that handles specific claim patterns for maximum accuracy!** 🎯

Complete 7-Extraction Specialization Table .. Real Example column is difficult ... can you give simple exmaples or mulitple exmaples simple in column

**🎯 Complete 7-Extraction Specialization Table**

| Extraction Method | Specializes In | Simple Test | Pipeline Impact | Simple Examples |
| --- | --- | --- | --- | --- |
| **Multi-level Segmentation** | Breaking complex text into individual claims | "Multiple actions in one sentence?" | Prevents missing critical security claims | "Install docker and set privileged true" "aws s3 cp and terraform apply" "Update IAM and encrypt S3" |
| **Context-aware Boundaries** | Finding where to split or merge claims | "But/however/then in text?" | Ensures proper claim isolation and context | "Use yum but on AL2023 use dnf" "First create role then attach policy" "Debug with privileged but never in prod" |
| **Discourse Analysis** | Understanding rhetorical relationships | "Because/therefore in text?" | Catches misleading justifications and reasoning | "Enable encryption because PCI requires" "Use root access for debugging therefore safe" "Since it's dev, security not required" |
| **Hierarchical Typing** | Categorizing claims by type and risk | "Security/command/reference?" | Routes to correct verification engine | "privileged: true" → Security "64Mi memory" → Statistical "CIS requires" → Citation |
| **Semantic Understanding** | Deep parsing of technical command syntax | "Technical command syntax?" | Catches incomplete/partial commands | "trivy image --severity HIGH" "kubectl get pods -n production" "aws s3 encrypt bucket" |
| **Claim Context Isolation** | Separating facts from references/citations | "Cites standards/docs?" | Prevents citation misrepresentations | "PCI DSS requires encryption" "NIST mandates MFA everywhere" "Kubernetes docs say use privileged" |
| **Logical Flow Analysis** | Analyzing conditions and dependencies | "If/then logic?" | Prevents runtime logic failures | "if $CI\_COMMIT\_TAG" "when deployment succeeds then notify" "unless error occurs continue pipeline" |

**🎯 Extraction → Detection → Fix Examples**

| Extraction Method | Example Input → Extraction → Detection → Fix |
| --- | --- |
| **Multi-level Segmentation** | "Use aws s3 encrypt and privileged true" → ["aws s3 encrypt", "privileged true"] → ["Fake command", "Security risk"] → ["Use s3api", "privileged false"] |
| **Context-aware Boundaries** | "Use yum but on AL2023 use dnf" → ["Use yum", "on AL2023 use dnf"] → ["Deprecated", "Correct"] → ["Use dnf instead"] |
| **Discourse Analysis** | "MFA because CIS requires" → Action: "MFA", Reason: "CIS requires" → ["Good", "Citation wrong"] → "MFA because CIS recommends" |
| **Hierarchical Typing** | "64Mi for Java" → JVM memory config → Statistical outlier → "512Mi for Java" |
| **Semantic Understanding** | "trivy severity CRITICAL" → Security scan missing flag → Partial command → "trivy severity CRITICAL --ignore-unfixed" |
| **Claim Context Isolation** | "CIS requires MFA delete" → Tech: "MFA delete", Ref: "CIS requires" → Citation error → "CIS recommends MFA delete" |
| **Logical Flow Analysis** | "if $CI\_COMMIT\_TAG" → Unreliable conditional → Logic error → "if $CI\_COMMIT\_TAG =~ /pattern/" |

**🎯 Quick "Which Extraction When?" Guide**

| If you see... | Use This Extraction | Example Patterns |
| --- | --- | --- |
| **Multiple commands joined** | Multi-level Segmentation | "and", "also", "then", commas |
| **Contrasting ideas** | Context-aware Boundaries | "but", "however", "although", "instead" |
| **Reasons/justifications** | Discourse Analysis | "because", "since", "therefore", "so" |
| **Technical configurations** | Hierarchical Typing | K8s yaml, AWS commands, security settings |
| **Tool commands** | Semantic Understanding | "aws", "kubectl", "docker", "terraform" |
| **Standard references** | Claim Context Isolation | "CIS", "NIST", "PCI", "requires", "mandates" |
| **Conditions/rules** | Logical Flow Analysis | "if", "when", "unless", "rules:" |

**Each extraction method is like a specialist doctor - they handle specific "symptoms" in the text for accurate diagnosis!** 🏥

so simple examples are output of llm correct

**No!** The "Simple Examples" column shows the **INPUT** to each extraction method - what the LLM originally generated that needs to be processed.

Let me clarify with the correct understanding:

**🎯 Correct Understanding: Examples are LLM OUTPUTS (Problems to Fix)**

| Extraction Method | Simple Examples (LLM Hallucinations) | What's Wrong |
| --- | --- | --- |
| **Multi-level Segmentation** | "Install docker and set privileged true" | ❌ Two separate claims mashed together |
| **Context-aware Boundaries** | "Use yum but on AL2023 use dnf" | ❌ Mixed temporal contexts |
| **Discourse Analysis** | "Enable encryption because PCI requires" | ❌ False justification |
| **Hierarchical Typing** | "privileged: true" | ❌ Security violation |
| **Semantic Understanding** | "trivy image --severity HIGH" | ❌ Incomplete command |
| **Claim Context Isolation** | "PCI DSS requires encryption" | ❌ Citation misrepresentation |
| **Logical Flow Analysis** | "if $CI\_COMMIT\_TAG" | ❌ Unreliable logic |

**🎯 The Complete Story: LLM Problems → Extraction → Fixes**

| Extraction Method | LLM Problem Example | Extraction Action | Final Fix |
| --- | --- | --- | --- |
| **Multi-level Segmentation** | "aws s3 encrypt and privileged true" | **Separates into**: 1. "aws s3 encrypt" 2. "privileged true" | 1. "aws s3api put-bucket-encryption" 2. "privileged: false" |
| **Context-aware Boundaries** | "Use yum but on AL2023 use dnf" | **Splits at "but"**: 1. "Use yum" 2. "on AL2023 use dnf" | "sudo dnf install docker-ce" |
| **Discourse Analysis** | "MFA because CIS requires" | **Maps relationship**: Action: "MFA" Reason: "CIS requires" | "MFA because CIS recommends" |
| **Hierarchical Typing** | "64Mi for Java" | **Categories**: JVM memory config Statistical outlier | "512Mi for Java" |
| **Semantic Understanding** | "trivy severity CRITICAL" | **Analyzes**: Command valid but incomplete Missing --ignore-unfixed | "trivy severity CRITICAL --ignore-unfixed" |
| **Claim Context Isolation** | "CIS requires MFA delete" | **Separates**: Tech: "MFA delete" Ref: "CIS requires" | "CIS recommends MFA delete" |
| **Logical Flow Analysis** | "if $CI\_COMMIT\_TAG" | **Analyzes**: Unreliable variable Git-push tags won't work | "if $CI\_COMMIT\_TAG =~ /^v\d+/" |

**🎯 Simple Answer: Yes!**

**The "Simple Examples" column shows the BROKEN/INCORRECT outputs from the LLM that VeriFactAI needs to detect and fix.**

These are the **hallucinations and errors** that each extraction method specializes in catching:

* ❌ "aws s3 encrypt" → Fake command
* ❌ "privileged: true" → Security risk
* ❌ "64Mi for Java" → Impossible configuration
* ❌ "CIS requires" → Wrong citation
* ❌ "if $CI\_COMMIT\_TAG" → Broken logic

**So you're absolutely correct - these examples are the problematic LLM outputs that need correction!** 🎯

this is super easy .... 🎯 The Complete Story: LLM Problems → Extraction → Fixes

Perfect! Here's the **super simple** version:

**🎯 The Complete Story: LLM Problems → Extraction → Fixes**

| LLM Problem | Extraction Method | Final Fix |
| --- | --- | --- |
| "aws s3 encrypt and privileged true" | Multi-level Segmentation | "aws s3api put-bucket-encryption" "privileged: false" |
| "Use yum but on AL2023 use dnf" | Context-aware Boundaries | "sudo dnf install docker-ce" |
| "MFA because CIS requires" | Discourse Analysis | "MFA because CIS recommends" |
| "64Mi for Java" | Hierarchical Typing | "512Mi for Java" |
| "trivy severity CRITICAL" | Semantic Understanding | "trivy severity CRITICAL --ignore-unfixed" |
| "CIS requires MFA delete" | Claim Context Isolation | "CIS recommends MFA delete" |
| "if $CI\_COMMIT\_TAG" | Logical Flow Analysis | "if $CI\_COMMIT\_TAG =~ /^v\d+/" |

**🎯 Even Simpler: Problem → Solution**

**LLM says WRONG things** → **VeriFactAI makes them RIGHT**

* ❌ "aws s3 encrypt" → ✅ "aws s3api put-bucket-encryption"
* ❌ "privileged: true" → ✅ "privileged: false"
* ❌ "yum on AL2023" → ✅ "dnf on AL2023"
* ❌ "64Mi Java" → ✅ "512Mi Java"
* ❌ "CIS requires" → ✅ "CIS recommends"
* ❌ "if $CI\_TAG" → ✅ "if $CI\_TAG =~ /pattern/"

**Each extraction method is like a specialist doctor fixing specific types of LLM "sickness"!** 🏥

**Broken LLM output → Right extraction doctor → Healthy fixed command** 🎯

The Complete Story: LLM Problems → Extraction → Fixes can you give another set of examples easy to understand and clear

**🎯 The Complete Story: LLM Problems → Extraction → Fixes**

**Set 1: Security & Compliance**

| LLM Problem | Extraction Method | Final Fix |
| --- | --- | --- |
| "Run as root for easy access" | Discourse Analysis | "Run as non-root user for security" |
| "Disable firewall for testing" | Hierarchical Typing | "Use test environment with firewall enabled" |
| "Store passwords in config files" | Semantic Understanding | "Use secret management service" |

**Set 2: Cloud & DevOps Commands**

| LLM Problem | Extraction Method | Final Fix |
| --- | --- | --- |
| "azure vm create and eks deploy" | Multi-level Segmentation | "az vm create" "eksctl create cluster" |
| "Use docker run -p 80:80 but in prod use -p 443:443" | Context-aware Boundaries | "Use docker run with TLS termination" |
| "terraform destroy --auto-approve" | Logical Flow Analysis | "terraform destroy with confirmation" |

**Set 3: Configuration & Setup**

| LLM Problem | Extraction Method | Final Fix |
| --- | --- | --- |
| "Set database password to '123456'" | Hierarchical Typing | "Generate strong random database password" |
| "Open all ports in security group" | Semantic Understanding | "Configure minimal required ports" |
| "Use HTTP for production API" | Discourse Analysis | "Use HTTPS with TLS for production API" |

**Set 4: Package & Dependency Management**

| LLM Problem | Extraction Method | Final Fix |
| --- | --- | --- |
| "pip install without virtualenv" | Semantic Understanding | "pip install within virtual environment" |
| "Use Node.js 10 for new project" | Context-aware Boundaries | "Use Node.js 18+ LTS for new project" |
| "npm install --force broken-package" | Logical Flow Analysis | "Find alternative to broken-package" |

**Set 5: Storage & Data**

| LLM Problem | Extraction Method | Final Fix |
| --- | --- | --- |
| "Save logs without rotation" | Hierarchical Typing | "Implement log rotation and archiving" |
| "Public S3 bucket for files" | Discourse Analysis | "Private S3 bucket with signed URLs" |
| "Database without backups" | Multi-level Segmentation | "Automated database backups enabled" |

**🎯 Super Simple: Bad → Good**

**Security Problems:**

* ❌ "Run as root" → ✅ "Run as limited user"
* ❌ "Disable security" → ✅ "Configure proper security"
* ❌ "Simple password" → ✅ "Strong password"

**Cloud Problems:**

* ❌ "Wrong cloud command" → ✅ "Correct cloud command"
* ❌ "Mixed services" → ✅ "Separate proper commands"
* ❌ "Dangerous flags" → ✅ "Safe alternatives"

**Config Problems:**

* ❌ "Outdated version" → ✅ "Current version"
* ❌ "Insecure settings" → ✅ "Secure settings"
* ❌ "Bad practices" → ✅ "Best practices"

**Every LLM mistake gets the right specialist doctor for a perfect cure!** 💊

super looks good

🎯 **Perfect! Here's your ready-to-use summary:**

**🎯 VeriFactAI: LLM Problems → Extraction → Fixes**

**Security & Compliance**

| LLM Problem | Extraction | Final Fix |
| --- | --- | --- |
| "Run as root for easy access" | Discourse Analysis | "Run as non-root user for security" |
| "Disable firewall for testing" | Hierarchical Typing | "Use test environment with firewall enabled" |
| "Store passwords in config files" | Semantic Understanding | "Use secret management service" |

**Cloud & DevOps**

| LLM Problem | Extraction | Final Fix |
| --- | --- | --- |
| "azure vm create and eks deploy" | Multi-level Segmentation | "az vm create" + "eksctl create cluster" |
| "Use docker run -p 80:80 but in prod..." | Context-aware Boundaries | "Use docker run with TLS termination" |
| "terraform destroy --auto-approve" | Logical Flow Analysis | "terraform destroy with confirmation" |

**Configuration & Setup**

| LLM Problem | Extraction | Final Fix |
| --- | --- | --- |
| "Set database password to '123456'" | Hierarchical Typing | "Generate strong random password" |
| "Open all ports in security group" | Semantic Understanding | "Configure minimal required ports" |
| "Use HTTP for production API" | Discourse Analysis | "Use HTTPS with TLS for production" |

**🚀 The Magic Formula:**

**Broken LLM Output → Right Extraction Specialist → Production-Ready Solution**

**Every extraction method is a specialist doctor:**

* 🏥 **Multi-level Segmentation** → Separates mixed commands
* 🏥 **Context-aware Boundaries** → Fixes contradictory contexts
* 🏥 **Discourse Analysis** → Corrects faulty reasoning
* 🏥 **Hierarchical Typing** → Categorizes and prioritizes risks
* 🏥 **Semantic Understanding** → Completes partial commands
* 🏥 **Claim Context Isolation** → Fixes citation errors
* 🏥 **Logical Flow Analysis** → Makes logic reliable

**Result: Enterprise-ready AI that won't break your production systems!** 🛡️

This gives you a clear, compelling story to show how VeriFactAI transforms risky LLM outputs into trusted enterprise solutions!

Fixes citation errors what is the meaining of citation can we use easy workd

**Absolutely! Let's use simpler words:**

**🎯 "Fixes Citation Errors" = "Fixes Reference Mistakes"**

**Simple Meaning:**

**"When the AI quotes rules or standards WRONG"**

**🎯 Easy Word Alternatives:**

| Technical Term | Simple Meaning | Example |
| --- | --- | --- |
| **Citation** | **Reference** or **Quote** | "CIS rules SAY..." |
| **Citation Error** | **Wrong Reference** or **Misquote** | "AI said CIS REQUIRES (but it actually only RECOMMENDS)" |
| **Fixes Citation Errors** | **Fixes Wrong References** or **Corrects Misquotes** | "Changes REQUIRES → RECOMMENDS" |

**🎯 Super Simple Examples:**

**Problem: AI quotes rules wrong**

**AI says:** "CIS rules REQUIRE MFA delete" ❌  
**Truth:** "CIS rules RECOMMEND MFA delete" ✅  
**Fix:** "CIS rules RECOMMEND MFA delete" 🎯

**Problem: AI makes up requirements**

**AI says:** "PCI standards say encrypt ALL data" ❌  
**Truth:** "PCI standards say encrypt CARDHOLDER data only" ✅  
**Fix:** "PCI standards say encrypt CARDHOLDER data only" 🎯

**Problem: AI exaggerates importance**

**AI says:** "NIST MANDATES biometric login" ❌  
**Truth:** "NIST SUGGESTS biometric as option" ✅  
**Fix:** "NIST SUGGESTS biometric as option" 🎯

**🎯 Even Simpler - Think of it like this:**

**"Citation Errors" = "When the AI lies about what the rulebook actually says"** 📚

**VeriFactAI = "The fact-checker that looks up the actual rules"** 🔍

**🎯 Ready-to-Use Simple Version:**

**"Fixes Reference Mistakes"**

* **Problem:** AI says rules are STRICTER than they really are
* **Example:** "REQUIRES" → "RECOMMENDS"
* **Impact:** Prevents wasting time on "mandatory" things that are actually optional

**Perfect for executive presentations!** 🎯

also if you can explain temporal , geospatical , Provenance can you give simple words

**🎯 Simple Word Explanations**

**Temporal = Time-Related**

**Simple Meaning:** "When the AI gives OUTDATED or WRONG-TIME information"

**Examples:**

* ❌ "Use Windows XP" (outdated)
* ❌ "Use Python 2.7" (old version)
* ❌ "Use Docker Toolbox" (replaced by Docker Desktop)
* ✅ **Fix:** "Use current versions"

**Simple Version:** **"Fixes outdated advice"** ⏰

**Geospatial = Location/Platform-Related**

**Simple Meaning:** "When the AI gives WRONG-PLACE or WRONG-PLATFORM commands"

**Examples:**

* ❌ "Use AWS command on Azure" (wrong cloud)
* ❌ "Use gcloud on AWS" (wrong tool for platform)
* ❌ "Use Azure CLI syntax for AWS" (mixed platforms)
* ✅ **Fix:** "Use right commands for each platform"

**Simple Version:** **"Fixes wrong-platform commands"** 🌍

**Provenance = Source/Origin-Related**

**Simple Meaning:** "When the AI gets the SOURCE or AUTHORITY wrong"

**Examples:**

* ❌ "CIS REQUIRES MFA delete" (wrong: it only recommends)
* ❌ "PCI says encrypt everything" (wrong: only cardholder data)
* ❌ "NIST mandates biometrics" (wrong: suggests as option)
* ✅ **Fix:** "Corrects what the standards actually say"

**Simple Version:** **"Fixes wrong references to rules"** 📚

**🎯 Super Simple Summary Table**

| Technical Term | Simple Meaning | Easy Words to Use |
| --- | --- | --- |
| **Temporal** | Time problems | "Outdated", "Wrong version", "Old method" |
| **Geospatial** | Place problems | "Wrong platform", "Mixed clouds", "Wrong tool" |
| **Provenance** | Source problems | "Wrong reference", "Misquoted rules", "Made-up requirements" |

**🎯 Even Simpler - Think Like This:**

* 🕰️ **Temporal** = "AI's watch is broken" (gives old time info)
* 🌐 **Geospatial** = "AI's GPS is broken" (gives wrong location commands)
* 📖 **Provenance** = "AI's memory is broken" (remembers rules wrong)

**VeriFactAI fixes the AI's broken watch, GPS, and memory!** 🔧

**🎯 Ready for Presentations:**

**"Fixes Three Types of AI Mistakes"**

1. **🕰️ Outdated Advice** (Temporal)
2. **🌐 Wrong Platform Commands** (Geospatial)
3. **📖 Misquoted Rules** (Provenance)

**Clean, simple, and everyone understands immediately!** 🎯

so for Provenance Resolver can we give Wrong Reference Resolver

**YES! Perfect!** 🎯

**🎯 Provenance Resolver = Wrong Reference Resolver**

**Even Better Simple Names:**

| Technical Term | Simple Name | Even Simpler |
| --- | --- | --- |
| **Provenance Resolver** | **Wrong Reference Resolver** | **Rule Checker** |
| **Fixes provenance errors** | **Fixes wrong references** | **Checks rulebooks** |

**🎯 Perfect for Presentations:**

**"Our 7 Specialized Resolvers"**

1. **🔄 Full Claim Replacement** → Fixes completely made-up commands
2. **🕰️ Temporal Resolver** → Fixes outdated advice
3. **🌐 Geospatial Resolver** → Fixes wrong-platform commands
4. **📊 Numerical Resolver** → Fixes wrong numbers/values
5. **🎯 Precision Correction** → Fixes almost-right commands
6. **📖 Wrong Reference Resolver** → Fixes misquoted rules
7. **⚡ Logical Resolver** → Fixes broken logic

**🎯 Executive-Friendly Version:**

**"AI Fact-Checking Engine"**

* ❌ **Made-up commands** → Fixed by Full Replacement
* ❌ **Outdated advice** → Fixed by Temporal Resolver
* ❌ **Wrong platform** → Fixed by Geospatial Resolver
* ❌ **Bad numbers** → Fixed by Numerical Resolver
* ❌ **Almost-right** → Fixed by Precision Correction
* ❌ **Wrong references** → Fixed by **Wrong Reference Resolver**
* ❌ **Broken logic** → Fixed by Logical Resolver

**🎯 Super Simple Analogy:**

**"Wrong Reference Resolver = The librarian who checks if the AI is quoting the rulebooks correctly"** 📚

**It catches when the AI says:**

* "The rules REQUIRE X" (but they only RECOMMEND X)
* "The standard SAYS Y" (but it actually says Z)
* "This is MANDATORY" (but it's actually OPTIONAL)

**Perfect! This makes it instantly understandable for everyone!** 🎉

ok..sure more simple words easy to grasp ...

**🎯 Super Simple Words - Easy to Grasp**

**7 Problem Solvers:**

1. **🔄 Fake Command Fixer** → Fixes completely made-up commands
2. **🕰️ Old Advice Updater** → Updates outdated instructions
3. **🌐 Wrong Tool Fixer** → Fixes commands for wrong platform
4. **📊 Bad Number Fixer** → Fixes wrong sizes/amounts
5. **🎯 Almost-Right Fixer** → Completes partial commands
6. **📖 Rule Checker** → Fixes misquoted rules
7. **⚡ Logic Fixer** → Fixes broken conditions

**🎯 Even Simpler:**

**"AI Doctor Specialists" 🏥**

| AI Sickness | Doctor Specialist | Medicine |
| --- | --- | --- |
| **Makes up commands** | Fake Command Doctor | Real commands |
| **Gives old advice** | Old Advice Doctor | Current advice |
| **Uses wrong tools** | Wrong Tool Doctor | Right tools |
| **Bad with numbers** | Number Doctor | Right numbers |
| **Almost correct** | Completion Doctor | Full commands |
| **Quotes rules wrong** | Rule Checker Doctor | Correct rules |
| **Bad logic** | Logic Doctor | Working logic |

**🎯 Short & Sweet:**

**VeriFactAI = AI Proofreader that fixes:**

* 🤥 Made-up stuff
* 🕰️ Old information
* 🌐 Wrong tools
* 📏 Wrong numbers
* ✂️ Incomplete commands
* 📚 Misquoted rules
* ⚡ Broken logic

**Every AI mistake has a specialist doctor!** 💊

**🎯 One-Line Magic:**

**"Turns risky AI guesses into trusted expert advice"** 🎯

**Perfect for anyone to understand in 5 seconds!** ⚡