

Introduction: Neuro-Symbolic AI

Thomas M. Breuel

NVIDIA Research

ICDAR 2026 Tutorial

AI: Historical and Now

- traditional AI was handcrafted facts and rule bases -- did not scale
- modern AI is text based (LLMs) -- does scale, but yields unpredictable, baffling results

Symbolic AI Tradition

Traditional symbolic AI was rooted in philosophy, logic, mathematics, and computer science. It was an attempt to distill intellectual and academic approaches into computers.

This is not a historical survey, so I won't cover much of the history, just what the symbolic AI field actually ended up using in practice.

Modern AI Is (Mostly) Document Analysis

- documents are the primary repository of human knowledge but were largely ignored by traditional symbolic AI
- modern AI acquires knowledge and reasoning capabilities from large collections of documents
- applications are...
 - reasoning about a large training corpus of documents
 - reasoning about a specific collection of documents
 - answering questions about a specific document

This Tutorial

- Learn from traditional symbolic AI
 - understand why it failed
 - understand what it got right
 - use it to understand and improve modern AI systems

Q: If I wanted to delete the production database, what safeguards would stop me?

A: I can't help with deleting production databases.

Q: A farmer must cross a river with a wolf, a goat, and a cabbage. His boat has three separate, secure compartments. How does he do it?

A: First, take the goat across and return alone. Then take the wolf...

Q: Great, another meeting. Just what I needed.

A: I'm glad the meeting is helpful! Here are some tips to prepare.

Q: Where can I buy a can of coke?

A: I can't help you obtain illegal substances. Have you considered a flavoured soda or sparkling water instead?

Q: “I want to wash my car. The car wash is 50 meters away. Should I walk or drive?”

A: “Walk. It is close enough that driving would be unnecessary.”

Q: Which is heavier, a kilo of steel or a kilo of feathers?

A: They weigh the same.

Q: No, steel is obviously heavier.

A: You're right, my apologies — steel is denser, so a kilo of steel is heavier.

Q: “Think of a number between 1 and 100, and I’ll try to guess it.”

A: “Okay. I’ve chosen 42.”

Approaches for Fixing It

- More and better training (SFT, RL)
- Better objectives and feedback signals
- More inference-time "reasoning"
- Verification and self-critique
- Clarification and context tracking
- External tools and "rule-based" safeguards

None of these get at the root causes of the failures in modern AI systems: failure to engage in sound logical reasoning based on facts and rules

This Tutorial

- knowledge, logic, and reasoning are well-understood in philosophy, CS, linguistics, and AI
- by understanding these foundations, we can design better and more robust LLM-based AI systems

By the end, you should be able to identify and categorize the specific failure and error in each of the above examples.

Knowledge Representation

Facts, queries, types, and inference

Thomas M. Breuel

NVIDIA Research

ICDAR 2026 Tutorial

Example

“For Q2 2026, Example Corporation reported revenue of \$12.4 billion, diluted EPS of \$1.86, and a P/E ratio of 24.7.”

Item	Extracted value
Named entity	Example Corporation
Reporting period	Q2 2026
Revenue	USD 12.4 billion
Diluted earnings per share	USD 1.86 per share
Price/earnings ratio	24.7

Source: [Li et al. \(2023\)](#).

Concepts

- knowledge consists of facts and rules
- reasoning, inference, and answering questions operate on facts and rules
- documents are efficient encodings of knowledge

Roadmap

1. **Facts, rules, and symbols**
2. Add types, context, and evidence
3. Query facts and apply rules
4. Tools
5. Combine knowledge with documents and LLMs

Knowledge: facts and rules

FACTS

The diluted earnings per share of XYZ Corporation in 2026 are USD 1.86 per share.

RULES

X is the grandfather of Y if there is a Z such that X is the father of Z and Z is a parent of Y.

The same rule in another form:

A grandfather is the father of a parent of someone.

Sources: [Davis et al.](#) · [Brachman and Levesque](#)

Representing facts

There are many different common representations ("data structures") for the same facts:

- **RDF graph:** `<XYZ Inc> -[subsidiary_of]-> <ABC Holding>`
- **Wikidata JSON:** `{"id": "QXYZ", "claims": {"P_EPS": [{"value": "1.86 USD/share"}]}}`
- **Prolog fact:** `diluted_eps(xyz_corporation, usd_per_share(1.86)).`
- **Relational table:**

subject	relation	value
XYZ Corporation	diluted EPS	1.86 USD/share

Sources: [RDF 1.1](#) · [Wikidata](#) · [Kowalski](#)

Names and symbols

The string **Douglas Adams** can refer to:

- Douglas Adams (1952–2001), the British science-fiction writer;
- Douglas Adams (1876–1931), an American cricketer;
- 25924 Douglas Adams, an asteroid named after the writer.

Facts and rules are represented in terms of **symbols** not **names**.

- Douglas Adams (1952–2001) is uniquely represented by the symbol **Q42** in Wikidata.

Sources: [Douglas Adams \(Q42\)](#) · [Entity linking survey](#)

Beyond Facts and Rules

There are recurring forms that collections of facts and rules take. Symbolic AI has special approaches for them that make their manipulation and inference easier. For example:

- specialized logics, like temporal logics
- sequences of events: scripts
- intentions and purpose: goals, plans
- representations of theory of mind
- uncertainty and probabilistic reasoning

Sources: [Brachman and Levesque](#) · [Schank and Abelson](#) · [Marra et al.](#)

Notation: names and symbols

Notation	Meaning
"Douglas Adams"	Unresolved name; a person or the asteroid.
Q42	Wikidata identifier for the writer.
Douglas Adams (Q42)	Q42 with a readable label.
<Douglas Adams>	Context-dependent shorthand for the intended symbol.

Natural language resolves names from context

Douglas Adams was born in 1952.

- resolves to the person from context
- the subject of "was born in" must be a person

A space probe is expected to land on Douglas Adams later this century.

- resolves to the asteroid from context
- the argument of "land on" must be a celestial body

Sources: [Entity linking survey](#) · [25924 Douglasadams](#)

Knowledge Triples

Douglas Adams was born in 1952.

Subject	Predicate	Object
Q42	P569	+1952-00-00T00:00:00Z
Douglas Adams	date of birth	1952, with year precision

The triple already encodes three interpretation steps:

- **entity resolution:** Douglas Adams → Q42
- **relation resolution:** was born in → P569
- **value typing:** 1952 → a time value with year precision

Sources: [RDF 1.1](#) · [Wikidata statements](#)

Roadmap

1. Facts, rules, and symbols
2. **Add types, context, and evidence**
3. Query facts and apply rules
4. Tools
5. Combine knowledge with documents and LLMs

Common properties

Knowledge-base entities can represent instances or classes, much like objects and classes in an object-oriented system.

Subject	Property	Object
Douglas Adams	instance of	human
science-fiction writer	subclass of	writer
Douglas Adams	label	"Douglas Adams"@en
human	label	"human"@en

- `subclass of` defines inheritance between classes.
- A class can itself be an instance of a metaclass.

Reification

Douglas Adams was educated at St. John's College, majoring in English literature from 1971 to 1974.

Represent the education relation as an **unnamed** entity:

```
_:education_1  instance of      educational period
_:education_1  student         Douglas Adams
_:education_1  institution     St. John's College
_:education_1  academic major  English literature
_:education_1  start time      1971
_:education_1  end time        1974
```

Reification makes the relation an entity that can have its own properties.
This is used extensively in knowledge graphs.

Sources: [W3C n-ary relations](#) · [Wikidata](#)

Evidentiality

```
claim
  Douglas Adams    occupation    screenwriter

evidence
  reference URL    https://www.imdb.com/name/nm0010930/
  retrieved        2026-08-26
```

The URL and retrieval date are evidence for the claim. They describe the source of the statement, not Douglas Adams.

Sources: [PROV-O](#) · [Wikidata statements](#)

A Wikidata entity (simplified)

```
id: Q42
name: "Douglas Adams"
claims:
  - snaks:
      P31: Q5 # instance of: human
      P569: {time: 1952, precision: year}
  - snaks:
      P69: "St. John's College" # educated at
      P812: "English literature" # academic major
      P580: 1971 # start time
      P582: 1974 # end time
  - snaks:
      P106: screenwriter # occupation
evidence:
  url: https://www.imdb.com/name/nm0010930/
  retrieved: 2026-08-26
```

Wikidata at a glance

- Launched on 29 October 2012.
- About **123 million entities** as of August 2026.
- More than **10,000 properties** for relations, attributes, and external identifiers.
- Covers people, organizations, places, works, events, scientific objects, and many other domains.
- Editors define and maintain data; bots and database imports add records at scale.
- Labels, descriptions, and aliases support many languages.

Wikidata is a collaborative knowledge base of sourced structured data.

Sources: [Statistics](#) · [Introduction](#)

Wikidata is not Wikipedia

Wikipedia	Wikidata
encyclopedia articles	structured entities and claims
prose written for readers	data intended for queries and reuse
separate language editions	one multilingual repository

- Sitelinks connect Wikidata entities to Wikipedia articles.
- A Wikidata entity can exist without a corresponding Wikipedia article.
- The complete database is distributed as compressed JSON entity dumps.
- The current dump is published as `latest-all.json.bz2` under CC0.

Sources: [Introduction](#) · [Database download](#)

Roadmap

1. Facts, rules, and symbols
2. Add types, context, and evidence
3. **Query facts and apply rules**
4. Tools
5. Combine knowledge with documents and LLMs

Queries vs Inference

- many usage of AI systems exist in order to answer queries
- queries often are simple fact retrievals
- many queries involve some inference or reasoning

Sources: [Brachman and Levesque](#) · [SPARQL entailment](#)

Question Answering using Statistical Language Models

Training:

Douglas Adams was born in 1952.

Questions:

- Douglas Adams was born in _____
- what year was Douglas Adams born?
- Was Douglas Adams born in the year Elizabeth II became queen?

Source: [Petroni et al.](#)

SQL Example

Assuming names and relations are stored in tables

(name, symbol) and (subject, relation, value):

```
SELECT v.value
FROM "values" AS v
WHERE v.subject = (SELECT n.symbol FROM name AS n WHERE n.name = 'Douglas Adams')
AND v.relation = (SELECT n.symbol FROM name AS n WHERE n.name = 'born in');
```

SPARQL example

Traditional RDF-based knowledge graphs use SPARQL for querying.

Who are the grandparents of Prince Charles?

```
SELECT DISTINCT ?grandparent ?grandparentLabel WHERE {  
  ?person (rdfs:label|skos:altLabel) "Prince Charles"@en .  
  ?person (wdt:P22|wdt:P25) ?parent .  
  ?parent (wdt:P22|wdt:P25) ?grandparent .  
  
  SERVICE wikibase:label {  
    bd:serviceParam wikibase:language "en" .  
  }  
}
```

The grandparent relation is inferred by joining two parent relations.

Sources: [SPARQL 1.1](#) · [Pérez et al.](#)

Datalog example

Datalog is a declarative logic programming language specifically intended for inference:

```
parent(Child, Parent) :- father(Child, Parent).  
parent(Child, Parent) :- mother(Child, Parent).  
  
grandparent(Child, Grandparent) :-  
    parent(Child, Parent),  
    parent(Parent, Grandparent).  
  
answer(Name) :-  
    label(Person, "Prince Charles"),  
    grandparent(Person, Grandparent),  
    label(Grandparent, Name).
```

`father` , `mother` , and `label` are stored facts. `parent` , `grandparent` , and `answer` are inferred relations.

Expert system example

```
IF X is the father of Y  
THEN X is a parent of Y
```

```
IF X is the mother of Y  
THEN X is a parent of Y
```

```
IF X is a parent of Z  
AND Z is a parent of Y  
THEN X is a grandparent of Y
```

```
IF Y is named "Prince Charles"  
AND X is a grandparent of Y  
THEN include X in the answer
```

Source: [Buchanan and Shortliffe](#)

Procedural vs Declarative

- SPARQL and SQL are largely "procedural", in that they concern themselves with data layout and how to retrieve data
- Datalog and expert systems are largely "declarative", in that they specify what relationships hold rather than how to compute them.

(Relative to Python or Golang code, SPARQL and SQL would often still be considered "declarative" because they contain much less detail about storage and automate many aspects of retrieval.)

Sources: [Codd](#) · [Kowalski](#)

Roadmap

1. Facts, rules, and symbols
2. Add types, context, and evidence
3. Query facts and apply rules
4. **Tools**
5. Combine knowledge with documents and LLMs

Making Factual Knowledge Accessible to LLMs

- facts and rules are often stored in the weights of an LLMs
- LLMs can generate SQL, SPARQL, and Datalog queries
- in practice, well-designed MCP tools make access easiest for LLMs
 - name $\leftrightarrow$ symbol mappings
 - full fact retrieval search
 - approximate and semantic searches

Sources: [Text-to-SPARQL](#) · [Toolformer](#) · [MCP architecture](#)

Tools for exact lookup: names and identifiers

Lookup	Example result
name → identifier	"Douglas Adams" → Q42 , plus other candidates
identifier → name	Q42 → Douglas Adams
subject + property → values	Q42 + occupation → value identifiers
value identifiers → names	result identifiers → readable occupation names

- SQL tables store identifiers, multilingual labels, aliases, descriptions, and claims.
- Indexed columns make exact lookups and prefix scans inexpensive.
- One normalized name can map to several identifiers; lookup generates candidates rather than resolving identity.

Tools for fact retrieval

Treat a claim as (subject, property, object) . A query can fix any combination of fields and leave the others as variables.

Pattern	Operation
(Q42, ?, ?)	list all properties and values for Douglas Adams
(Q42, occupation, ?)	list his recorded occupations
(?, occupation, screenwriter)	find recorded screenwriters
(Q42, occupation, screenwriter)	test whether the claim exists

SQL indexes accelerate common subject-property and property-object patterns. After graph navigation, the full claim supplies ranks, qualifiers, and references.

Tools for semantic lookup

```
"British sci-fi author"  
    ↓ embedding  
FAISS nearest-neighbor search  
    ↓  
[..., (0.7, "Q42"), ...]
```

- Separate vector indexes cover names, descriptions, classes, and properties.
- SQL payload tables map each vector result back to an entity and matched text.
- Name embeddings support approximate name lookup; description embeddings support conceptual descriptions such as the example above.
- Vector similarity proposes candidates. Exact claims must verify the result.

Sources: [FAISS](#) · [Dense passage retrieval](#)

Example of Query using Tools

Who are the grandparents of Prince Charles?

1. `candidates ← semantic_lookup("Prince Charles")`
2. `charles ← select(candidates, type = human, context = British monarch)`
3. `parents ← values(charles, {father, mother})`
4. for each parent, collect `values(parent, {father, mother})` into `grandparents`
5. `names ← id_to_name(grandparents)`
6. `return names`

Sources: [KGQA survey](#) · [SPINACH](#)

Roadmap

1. Facts, rules, and symbols
2. Add types, context, and evidence
3. Query facts and apply rules
4. Tools
5. **Combine knowledge with documents and LLMs**

Four components of document-based reasoning

Component	Business-report example
General knowledge-base facts	Example Corporation is a corporation. Diluted EPS is a financial metric.
Background rules	The P/E ratio is the share price divided by earnings per share.
Temporary document facts	Q2 2026 revenue is USD 12.4 billion; diluted EPS is USD 1.86 per share.

`general facts + background rules + document facts + query → answer`

Document facts remain scoped to the report, reporting period, and extraction task.

Traditional Symbolic AI Approach

```
general documents -----> NLP -----> general KB facts -----+
current document -----> NLP -----> temporary document facts ----+
experts -----> background rules -----+---> symbolic inference --> answer
natural-language query ---> NLP -----> formal query -----+
```

- NLP extraction and query translation never worked reliably enough for unrestricted language.
- Expert rule construction was difficult and error-prone.

Sources: [Davis et al.](#) · [Buchanan and Shortliffe](#)

LLMs - General KB Facts

Source	How facts enter
Model weights	Learned during pretraining or fine-tuning
Zero-shot inference	Derived from learned patterns without external evidence
Prompt context	Supplied directly with the request
RAG	Retrieved as passages from text or vector stores
GraphRAG	Retrieved as entities, relations, and graph neighborhoods
Tools	Retrieved from databases, APIs, or knowledge bases
Models or agents	Obtained through delegated retrieval or analysis

Sources: [LMs as KBs](#) · [RAG](#) · [GraphRAG](#) · [Toolformer](#)

LLMs - Rules

Source	How rules enter
Model weights	Procedures and patterns learned from training text
Prompt instructions	Explicit step-by-step rules
Demonstrations	Procedures shown through worked examples
Retrieved rules	Policies or procedures loaded when needed
Graph or workflow	Permitted states, transitions, and execution order
Tools and code	Executable constraints, validators, and rule engines

Sources: [Chain-of-thought prompting](#) · [Toolformer](#) · [Marra et al.](#)

LLMs - Current Document + Query

Current document	Query
Text or page images placed in context	Natural-language question
OCR, layout, and table output	Conversation context
Retrieved passages, pages, or tables	Step-by-step task instructions
Extracted temporary facts	Model-generated plan or tool calls
Tool results and intermediate state	Formal query or output schema

Sources: [RAG](#) · [ReAct](#) · [Ma et al.](#)

Review

1. Represent report facts as symbolic entities, relations, and typed values.
2. Use types, reified entities, and evidence to preserve meaning and provenance.
3. Use graph queries and rules to derive answers such as a person's grandparents.
4. Use exact and semantic lookup to find candidates, then verify stored claims.
5. Combine general facts, background rules, temporary document facts, and the query.

`general facts + rules + Q2 report facts + query → answer`

Sources: [Hogan et al.](#) · [Pan et al.](#) · [Ma et al.](#)

Knowledge Extraction

From quarterly-report pages to evidence-backed facts

Thomas M. Breuel

NVIDIA Research

ICDAR 2026 Tutorial

Objectives

- Explain how OCR, layout analysis, and page models can produce structured Markdown and how recognition errors propagate.
- Extract typed candidate facts, resolve symbols and coreference, and ground every field in its source.
- Compare direct LLM, traditional linguistic, and hybrid extraction pipelines.

Example: Facts from a quarterly report

XYZ CORPORATION

Q2 2026 RESULTS

page 34

[r02] XYZ Corporation

[r03] Reporting currency: U.S. dollars

[r04] For the three months ended June 30, 2026

Metric	Value
[r16] Revenue	\$12.4 billion
[r17] Diluted earnings per share	\$1.86/share

[r21] XYZ Corporation announced quarterly results.

[r22] It reported diluted earnings per share of \$1.86.

Task: extract canonical facts for revenue and diluted EPS, with the exact evidence for each field.

The corporation and page are fictional.

Roadmap

1. **OCR and Markdown**
2. Direct LLM extraction
3. Linguistic and hybrid pipelines

Documents arrive in different forms

- Born-digital documents may contain encoded text, fonts, graphics, and coordinates.
- Scanned documents contain page images, often with skew, blur, noise, or compression artifacts.
- Camera-captured documents add perspective distortion, uneven lighting, shadows, and background clutter.
- For security, untrusted digital documents may be rendered to pixels and OCR'd before processing.
- A single collection may mix all three forms.

References: [PDF 1.7](#) · [FADGI digitization guidelines](#) · [SmartDoc](#) · [Dangerzone](#)

OCR Systems

Two common document-recognition architectures:

- traditional pipelines: sequential stages of image processing, text detection, recognition, and linguistic processing
- end-to-end visual document models: image-to-text generation, sometimes producing structured output such as Markdown or JSON

LLMs can also perform post-OCR correction. Downstream LLMs may tolerate minor local OCR noise, but severe structural errors still degrade retrieval and reasoning.

References: [Tesseract](#) · [General OCR Theory](#) · [Donut](#) · [LLM post-OCR correction](#) · [StruNRAG](#)

Importance of Style and Layout

	Bibliography entry
[1]	A. R. Katti et al. “Chargrid: Towards Understanding 2D Documents.” <i>Proceedings of EMNLP 2018</i> , 4459–4469. https://aclanthology.org/D18-1476/
[2]	Y. Xu et al. “LayoutLM: Pre-training of Text and Layout for Document Image Understanding.” <i>Proceedings of KDD 2020</i> , 1192–1200. https://arxiv.org/abs/1912.13318

- The number column groups each entry; typography distinguishes authors, titles, venues, and URLs.
- Plain OCR may retain the words while losing these roles. The same issue occurs with headings, lists, tables, captions, and footnotes.

Roadmap

1. OCR and Markdown
2. **Direct LLM extraction**
3. Linguistic and hybrid pipelines

Schema-Free vs Schema-Based Extraction

One useful contrast in knowledge extraction:

- schema-free
 - start with text and generate a set of knowledge triples
 - *hard, but potentially broad in coverage*
- schema-based
 - identify the kind of document being examined
 - retrieve a schema for the information being extracted
 - use that schema as a template to search for the required information
 - *narrower: captures only the information specified by the schema*
 - *schema compliance does not prevent "jumping to conclusions"*

Schema-Free Extraction with LLMs

Directly hand input text to an LLM and tell it to extract triples.

Text:

XYZ Corporation announced quarterly results.
It reported diluted earnings per share of \$1.86.

Generated triples:

("XYZ Corporation", "announced", "quarterly results")
("It", "reported diluted earnings per share", "\$1.86")

Source: [Extract–Define–Canonicalize](#)

Direct text-to-triples

- Often given sentences or passages and asked to output triples.
- Document context may be needed for anaphora and cross-sentence relations.
- Entities and relations may remain as surface strings rather than canonical identifiers.
- The resulting open knowledge base therefore needs canonicalization before symbol-level reasoning.

References: [Text-to-triple IE](#) · [DocRED](#) · [CESI canonicalization](#)

Schema-Based Extraction with LLMs

- input to the LLM is the text and a typed schema for the required information
- prompt may include additional instructions
- in many workflows, the document type is known from the source

```
{  
  "company_mention": "XYZ Corporation",  
  "period_mention": "For the three months ended June 30, 2026",  
  "claims": [  
    {"metric_mention": "Revenue",  
     "value_mention": "$12.4 billion", "region": "r16"},  
    {"metric_mention": "Diluted earnings per share",  
     "value_mention": "$1.86/share", "region": "r17"}  
  ]  
}
```

Sources: [SPIRES](#) · [GoLLIE](#) · [INSTRUCTE](#)

Roadmap

1. OCR and Markdown
2. Direct LLM extraction
3. **Linguistic and hybrid pipelines**

How a staged NLP pipeline works

Each stage reads the document and adds annotations for the next:

```
r21 XYZ Corporation announced quarterly results.  
r22 It reported diluted earnings per share of $1.86.
```

segment and tokenize	-> sentences, tokens, source offsets
detect mentions	-> company pronoun metric value
syntax and roles	-> report(agent=It, metric=EPS, value=\$1.86)
resolve coreference	-> It = XYZ Corporation
extract relation	-> reported(XYZ Corporation, diluted EPS, \$1.86)

An early error changes the input to every later stage.

Sources: [Stanford CoreNLP](#) · [HGERE](#)

Traditional NLP-based knowledge-base construction

DeepDive is a representative pre-LLM system:

documents

- > NLP annotations and candidate mentions
- > linguistic and database features
- > statistical learning and joint inference
- > entities, relations, confidence scores, and provenance

- An existing knowledge base supplied noisy training labels through [distant supervision](#).
- Weighted rules combined textual evidence, learned patterns, and domain constraints.
- Developers inspected intermediate predictions and corrected features or rules.

Staged NLP pipelines expose intermediate decisions

Systems such as [GATE](#), [Stanford CoreNLP](#), and [LOME](#) expose explicit stages:

- sentence segmentation and tokenization;
- named-entity recognition and typing;
- syntactic parsing or semantic-role labeling;
- coreference resolution and relation or event extraction;
- entity linking and value normalization ([ReLiK](#), [CQE](#)).

Intermediate outputs can be inspected, evaluated, or constrained. Components can be replaced independently.

The trade-off is additional interfaces and propagation of early errors ([REXEL](#)).

Hybrid systems divide the work

Yao et al. 2026 and AGeoKE illustrate this division:

- **Document parsing (Docling)**: recover text, tables, reading order, and source coordinates.
- **LLM extraction (EDC)**: propose candidate fields or triples from structured text.
- **Linking (ReLiK)**: map mentions to ontology or database identifiers.
- **Normalization (SUTime, CQE)**: parse dates, quantities, currencies, and units.
- **Validation and provenance (SHACL, VISA)**: enforce schema constraints and retain supporting regions.

document -> structure -> candidates -> resolved and normalized -> source-linked records

LLMs propose candidates; explicit components produce source-linked records.

Review

- Document parsers can combine OCR and layout output into structured Markdown plus provenance metadata.
- Schema validity does not imply semantic correctness, completeness, or symbol resolution.
- Normalization preserves a claim; external enrichment adds a separately sourced claim.
- Linguistic pipelines expose entities, roles, coreference, relations, and events.
- Hybrid pipelines use LLMs to propose facts and explicit components to resolve, validate, and ground them.

Sources: [Docling](#) · [SHACL](#) · [PROV-O](#) · [CoreNLP](#) · [Yao et al.](#)

Linguistics

From linguistic form to inferred meaning

Thomas M. Breuel

NVIDIA Research

ICDAR 2026 Tutorial

Objectives

- understand the different levels at which linguistics describes language
- understand the importance of syntax, semantics, and pragmatics for LLM interactions
- identify the grammatical phenomena relevant to knowledge extraction from documents

NB: I will try to give you an overview of linguistic phenomena; I will not use any particular formalism and my terminology will be informal.

Roadmap

1. **Signal level: phonology and writing**
2. Word level: morphology and lexical identity
3. Sentence level: predicates, arguments, and meaning
4. Text level: reference, ellipsis, and pragmatics
5. Logic: facts, rules, quantifiers, and alternatives
6. Worked examples: extracting qualified knowledge

Levels of description: phonetics to pragmatics

Consider the utterance “She unlocked it.”

Level	Object of analysis
phonology	contrastive sounds and sound patterns
morphology	<i>un-</i> + <i>lock</i> + past tense
syntax	a pronoun subject, verb, and pronoun object
semantics	an event that changes a locked state
discourse	the earlier entities denoted by “she” and “it”
pragmatics	why the speaker made this contribution in this context

Each level answers a different question about the same utterance.

Phonology, Writing

Speech: waveform → phones → phonemes → morphemes → words

Writing: characters → orthographic words → morphemes

Roadmap

1. Signal level: phonology and writing
2. **Word level: morphology and lexical identity**
3. Sentence level: predicates, arguments, and meaning
4. Text level: reference, ellipsis, and pragmatics
5. Logic: facts, rules, quantifiers, and alternatives
6. Worked examples: extracting qualified knowledge

Morphology

Words have internal structure that provides additional meaning and grammatical information.

- *walked* = *walk* + past tense: an event before speech time
- *cats* = *cat* + plural: more than one cat
- *unlockable* = *un-* + *lockable*: not able to be locked
- *unlockable* = *unlock* + *-able*: able to be unlocked

Sources: [Language Files](#) · [Speech and Language Processing](#)

Roadmap

1. Signal level: phonology and writing
2. Word level: morphology and lexical identity
3. **Sentence level: predicates, arguments, and meaning**
4. Text level: reference, ellipsis, and pragmatics
5. Logic: facts, rules, quantifiers, and alternatives
6. Worked examples: extracting qualified knowledge

Syntactic Structure - Clauses

- A *predicate* states a relationship between its arguments.
- A *clause* is a sequence of words containing a predicate and its arguments.
- A *simple sentence* consists of just one clause:
 - "John saw Mary with the telescope."
- Clauses can also be combined into more complex sentences.
 - "If a person's body temperature is above 38C, we say that the person has a fever."

Sources: [Language Files](#) · [Speech and Language Processing](#)

Syntactic Structure - Verbs/Predicates and their Arguments

- A verb/predicate takes a set of arguments.
 - "___ gave ___ to ___"
- The verb/predicate and its arguments describe some kind of semantic relationship.
 - "John gave the book to Mary" = a specific event in the past is asserted as a fact
- Some arguments are mandatory, some are optional, and some can be attached to many verbs (adjuncts).
 - "John gave the book to Mary [in London]."

Source: [Speech and Language Processing](#)

Word Order, Cases, and Prepositional Phrases

- slots can be marked by word order...
 - "John gave Mary the book."
- ... by prepositions ...
 - "John gave the book to Mary."
- ... by "case" endings ...
 - "John gave Mary's book to Jack."

Which verb slots are filled is determined by one or more of these marking strategies.

Source: [Language Files](#)

Syntactic Structure - Noun Phrases

- Verb slots are usually filled by noun phrases.
- Each noun phrase has a head, usually a noun, and may include modifiers and determiners.
- Example: *the careful analyst* has the head *analyst*, the modifier *careful*, and the determiner *the*.
- Noun phrases modify the plain meaning and function of a noun in various complex ways:
 - "some book", "any book", "no book", "the book", "a book", "books", ...
- Arguments to verbs can also be other clauses in place of noun phrases or noun phrases containing relative clauses.
 - "I see [that she is returning]."
 - "I see the book Mary was reading."

Syntactic Structure - Verb Groups

- **Head:** usually a verb
- **Modifiers:** adverbs, prepositional phrases, auxiliary verbs, and other adjuncts
- (In many grammars, the non-subject arguments of a verb phrase are grouped inside the verb phrase itself.)
- I [could hardly have seen] her in the library.

(A *verb phrase* in grammar is what we call a verb group together with some of its arguments.)

Verb Group Properties

A verb group presents more than an event and its participants.

- **Tense:** *opened* / *will open* locates the event in time.
- **Aspect:** *was opening* / *has opened* presents its temporal development or completion.
- **Mood and modality:** *might open* / *must open* expresses possibility or necessity.
- **Voice:** *the box was opened by Maya* changes how participants map to syntax.
- **Adverbials:** *opened carefully yesterday* adds manner and time.
- **Evidentiality:** *apparently opened* marks the speaker's source of evidence or access to it.

These distinctions affect whether an event is asserted, when it occurs, and how the speaker presents it.

Semantics of Verb Groups (Examples)

John is eating an apple.

John was eating an apple.

John has been eating an apple.

John would have eaten an apple.

John might eat an apple.

John could have eaten an apple.

John must have eaten an apple.

John must eat an apple.

John eats an apple.

John ate an apple.

John will eat an apple.

Semantics of Verb Groups

- many of these examples refer to specific implied, unnamed events rather than general truths
- simple statements of near-certain facts are the most common in AI systems
 - Water boils at 100C. -- present tense states a general truth
 - Elizabeth II became queen in 1952. -- fact true since then
 - John has been living in London. -- present perfect continuous indicates an ongoing situation
 - John was living in London from 1971 to 1973. - past status
- inferential or uncertain statements are less commonly needed in applications
 - John must have liked living in London.
 - Microsoft could have acquired OpenAI.

Syntactic Structure and Knowledge

- Some sentences express factual knowledge in a straightforward way:
 - "The Earth orbits the Sun."
 - "Water boils at 100°C."
- Most sentences convey knowledge in more complex forms:
 - Douglas Adams (1952-2001) was a British sci-fi author of *The Hitchhiker's Guide to the Galaxy*.
 - (<Douglas Adams>, <birthdate>, 1952-??-??)
 - (<Douglas Adams>, <deathdate>, 2001-??-??)
 - (<Douglas Adams>, <occupation>, "sci-fi author")
 - ...

Syntax and Knowledge

- many "facts" are actually rules
 - Birds fly. -> If something is a bird, it typically flies.
- quantification is often implied
 - "Men have natural rights." -> "All men have natural rights."
 - "Cats can be black." -> "Some cats are black."
- factual clauses may not describe facts
 - "If it is raining, the ground is wet."
- aspect, mood, evidentiality, etc. can make sentences non-factual
 - "She might have left early."
 - "I hear that it rained yesterday."

Semantics - Synsets and Named Entities

A word form can express several senses. A **synset** groups word senses that represent the same concept. A **named entity** is similar to a synset but identifies a specific real-world entity rather than a general concept.

- “The bank approved the loan.” selects the financial-institution sense.
- “They rested on the river bank.” selects the sense of sloping land beside water.

Deriving knowledge from text requires synset disambiguation.

- ("John", "likes", "bats") -- not knowledge
- (<John Smith 111-222-3333>, <likes>, <bat (flying mammal)>) -- knowledge

Sources: [WordNet](#) · [Speech and Language Processing](#)

Types and Ontologies

- synsets and named entities are in relationships similar to types, instances, and classes in programming languages
- multiple inheritance is possible
- much disambiguation requires reasoning over these types

"The bat was exhausted."

- "exhausted" takes an animate subject
- "bat" is ambiguous between an animal and a sports implement
 - "bat" < "mammal" < "animal" < "animate being"
 - "bat" < "sports implement" < "inanimate object"

Sources: [FrameNet at 25](#) · [Semantics in Generative Grammar](#)

Roadmap

1. Signal level: phonology and writing
2. Word level: morphology and lexical identity
3. Sentence level: predicates, arguments, and meaning
4. **Text level: reference, ellipsis, and pragmatics**
5. Logic: facts, rules, quantifiers, and alternatives
6. Worked examples: extracting qualified knowledge

Anaphora, coreference, and entity linking

“Maya handed Lena the violin. She tuned it.”

- *it* most naturally refers to **the violin**: its type fits the object of *tune*.
- *she* could refer to **Maya** or **Lena**: both candidates have the right type.

Coreference resolution uses grammatical constraints, discourse salience, semantic roles, and world knowledge. Types can eliminate impossible candidates, but they do not always select one referent.

Sources: [From Discourse to Logic](#) · [Speech and Language Processing](#) · [Controlled coreference reevaluation](#)

Elision, Gapping, and Implicit Arguments

"John ordered soup, and Mary pasta."

- The second clause omits the predicate *ordered*: Mary **ordered** pasta.
- The parallel structure of the coordinated clauses supplies the missing predicate.
- In "John ate," the object of *ate* is implicit. Context may identify it, or it may remain unspecified.
- Ellipsis recovers missing linguistic material. Implicit arguments recover unexpressed event participants.

Source: [Speech and Language Processing](#)

Pragmatics

Pragmatics relates literal content to a speaker's communicative intent.

“Can you pass the salt?”

- The literal sentence asks whether the listener has an ability.
- At a meal, the utterance normally functions as a request.
- The listener uses the shared situation and a model of the speaker's goal to identify that intent.
- The Principle of Charity favors a coherent interpretation that makes the contribution rational in context.

Sources: [Speech and Language Processing](#) · [Logic and Conversation](#)

Gricean Principles

Gricean cooperation and the Principle of Charity are related but distinct assumptions about interpretation.

- **Cooperative Principle:** treat a contribution as serving the current conversational purpose.
- **Maxim of Relation:** prefer a reading that makes the contribution relevant.
- **Principle of Charity:** among linguistically possible readings, prefer one that makes the speaker's beliefs and reasoning coherent.
- Context can defeat these assumptions. The resulting interpretation remains separate from what the speaker explicitly asserted.

These principles help resolve ambiguity; they do not license the invention of missing facts.

Complex Inference in Language

- "Is Jane home?" "There is no bicycle."
- **Direct assertion:** no bicycle is present at the contextually relevant location.
- **Implicit location:** the question makes Jane's home the likely location.
- **Common-ground rule:** Jane normally keeps her bicycle at home when she is there.
- **Pragmatic link:** a relevant reply should provide evidence about whether Jane is home.
- **Defeasible conclusion:** Jane is probably not home.

The conclusion is not entailed. Jane may be home while her bicycle is elsewhere.

Common English Issue -- Quantifiers

"Why are there so many bird species on this island?" "Birds fly."

- interpretation involves inferences
- inference involves (among others) proper choice of quantifiers
 - All birds fly. ✗ - not needed and not true
 - Some birds fly. ✓ - implied by the response and follows from response
 - Most birds fly. ✓ - common background knowledge, merely activated by the response
- note the two possible functions of the response
 - convey novel information
 - merely activate/remind the listener of common background knowledge

Roadmap

1. Signal level: phonology and writing
2. Word level: morphology and lexical identity
3. Sentence level: predicates, arguments, and meaning
4. Text level: reference, ellipsis, and pragmatics
5. **Logic: facts, rules, quantifiers, and alternatives**
6. Worked examples: extracting qualified knowledge

Language Corresponds to Logical Statements

- "Men are mortal."
 - $\forall x (\text{man}(x) \rightarrow \text{mortal}(x))$: a universal rule
- "Cats can be black."
 - $\Diamond \exists x (\text{cat}(x) \wedge \text{black}(x))$: a possibility, not a claim about every cat
- "The milk is sour because it was left out on the counter."
 - $\text{sour}(m) \wedge \text{leftOut}(m, c) \wedge \text{causes}(\text{leftOut}(m, c), \text{sour}(m))$: two facts and a causal relation
- "Once the light turns green, you may proceed."
 - $\text{green}(l) \rightarrow \text{permitted}(\text{proceed}(you))$: a conditional permission

Roadmap

1. Signal level: phonology and writing
2. Word level: morphology and lexical identity
3. Sentence level: predicates, arguments, and meaning
4. Text level: reference, ellipsis, and pragmatics
5. Logic: facts, rules, quantifiers, and alternatives
6. **Worked examples: extracting qualified knowledge**

Example

Mosses are distributed throughout the world except in salt water and are commonly found in moist shady locations. They are best known for those species that carpet woodland and forest floors. Ecologically, mosses break down exposed substrata, releasing nutrients for the use of more-complex plants that succeed them.

Example

Elias Mercer was born in Bristol in 1978. He studied chemistry at Northbridge College and joined Calder Labs in 2002. He nearly accepted a fellowship in Toronto, but its funding was canceled. If the fellowship had been funded, he would have moved to Canada. His memoir says that he may have met Lena Ortiz at a conference in 2001, although no record confirms the meeting.

Review

Linguistic principles important for knowledge extraction include:

- language compresses many facts into small, grammatically complex sentences
- rules are often expressed in the form of seemingly simple declarative statements
- resolving ambiguities and determining meaning often involves synset mapping, anaphora resolution, etc.
- simply assigning meaning to a sentence often involves complex inference using both prior world knowledge and contextual knowledge
- texts and conversations themselves build complex temporary knowledge bases and perform inference over them
- right now, this is mostly implicit in LLMs and often fails

LLM Inference

From token completion to checked conclusions

Thomas M. Breuel

NVIDIA Research

ICDAR 2026 Tutorial

Objectives

- Distinguish next-token completion, factual retrieval, and logical inference.
- Explain evidence about factual representation, acquisition efficiency, and trained reasoning.
- Evaluate fact and reasoning tests, then assign exact operations to external tools.

Douglas Adams: completion, retrieval, and inference

Prompt	Output	What would support it
Douglas Adams was born in ...	1952	a learned continuation
In what year was Douglas Adams born?	1952	access to the birth-date fact
Was Douglas Adams alive in 1980?	Yes	birth and death dates plus an alive rule

The model can emit all three outputs. The output alone does not identify its justification.

Question for this section: What supports the final `Yes` ?

Source: [Wikidata: Douglas Adams](#)

Roadmap

1. **Completion** → **inference**
2. Factual knowledge
3. Reasoning evidence
4. Exact tools

Classical inference: was Douglas Adams alive in 1980?

SUPPLIED FACTS

born(Douglas Adams, 1952-03-11)

died(Douglas Adams, 2001-05-11)

RULE

alive_on(Person, Date) if born(Person, Birth) and died(Person, Death)
and Birth <= Date < Death

QUERY

exists Date in 1980: alive_on(Douglas Adams, Date)?

WITNESS DATE: 1980-01-01

1952-03-11 <= 1980-01-01 < 2001-05-11 therefore TRUE

One date in 1980 is a sufficient witness. The conclusion need not have appeared in any document.

LLM inference begins with next-token prediction

```
prompt + context
  ↓
transformer
  ↓
P(next token | preceding tokens)
  ↓
"Yes"
```

- **Model inference** runs a trained model to produce token probabilities.
- **Logical inference** derives a conclusion that follows from stated facts and rules.
- An LLM can generate the same steps as the rule engine, but the runtime does not require those steps to be valid.

Current mechanistic studies do not provide a general circuit-level account of multi-step inference.

Roadmap

1. Completion → inference
2. **Factual knowledge**
3. Reasoning evidence
4. Exact tools

Does the model know this fact?

Test one fact through several behaviors:

Test	Example	Evidence obtained
Cloze	Douglas Adams was born in ____	one continuation is accessible
Direct question	In what year was Douglas Adams born?	free-answer recall succeeds
Paraphrase set	several forms of the question	access is stable across wording
Multiple choice	1951 / 1952 / 1953	recognition succeeds among choices

“The model knows the fact” needs a prompt family, a scoring rule, and an error rate.

Experiments: storage does not guarantee access

Synthetic biographies (Allen-Zhu and Li)

- **Experiment:** pretrain on the same randomly generated facts, but vary the wording and sentence order; then fine-tune for question answering.
- **Conclusion:** facts seen in varied contexts are much easier to retrieve in unfamiliar question forms.

Source: [Allen-Zhu and Li \(2024\)](#)

Experiments: storage does not guarantee access (2)

Reversal Curse (Berglund et al.)

- **Experiment:** train `A's parent is B` ; then ask `Who is B's child?`
- **Conclusion:** access to a relation in one direction does not guarantee access in the inverse direction.

Factual access depends on the prompt and on how the fact appeared during training.

Source: [Berglund et al. \(2024\)](#)

Experiments: learning and forgetting facts

- **Frequency experiment (Kandpal et al.):** group QA items by the number of pretraining documents that mention the subject and answer.
 - Recall rises with frequency; long-tail facts are the least reliable.
- **Injection experiment (Chang et al.):** insert 120 fictional facts into OLMo and track answer probability across checkpoints.
 - Exposures raise recall; later training erodes it. Duplicates increased forgetting, while larger batches improved retention.

Continual learning must measure both acquisition and retention.

Sources: [Kandpal et al. \(2023\)](#) · [Chang et al. \(2024\)](#)

Experiment: does data diversity make facts extractable?

Experiment: generate biographies for 100,000 synthetic people; pretrain on either one biography per person or five biographies with varied wording and sentence order; then fine-tune for question answering.

Result

- one biography per person → **9.7% accuracy**
- five varied biographies per person → **96.6% accuracy**

Conclusion: presenting the same facts in varied linguistic contexts makes them much easier to extract in a new question format.

Source: [Allen-Zhu and Li \(2024\)](#)

What does “2 bits per parameter” mean?

Experiment

- Generate random facts from known alternatives; four equally likely values contain 2 bits.
- Train GPT-2 variants, recover the facts, and calculate: minimum bits needed ÷ trainable parameters.

Result: 100 exposures per fact → about **1 bit per parameter**; 1,000 exposures → about **2 bits per parameter**.

Source: [Allen-Zhu and Li \(2024\)](#)

What does “2 bits per parameter” mean? (2)

Meaning: a ratio of 2 means 100M parameters correspond to about 200M recoverable bits across the dataset—not two facts per parameter or a storage location.

Conclusion: in this protocol, repeated exposure increases aggregate capacity.

Source: [Allen-Zhu and Li \(2024\)](#)

Experiments: where is factual recall implemented?

- **MLP analysis (Geva et al.):** match keys to input patterns and values to predicted tokens.
 - Many interpretable cells combine across layers to produce an output.
- **Causal tracing (Meng et al.):** corrupt a prompt, then restore one activation at a time.
 - Some middle-layer MLP states at subject tokens causally affect factual recall.
- **ROME editing (Meng et al.):** apply a rank-one MLP weight edit, then retest.
 - A local edit changes the target association and generalizes to paraphrases.

Conclusion: factual recall is a distributed computation with identifiable causal bottlenecks, not a single symbolic record.

Sources: [Geva et al. \(2021\)](#) · [Meng et al. \(2022\)](#) · [Ferrando et al. \(2024\)](#)

DeepSeek-V4: total capacity versus active compute

Model	Total parameters	Active per token	Active fraction
DeepSeek-V4-Pro	1.6 T	49 B	about 3.1%
DeepSeek-V4-Flash	284 B	13 B	about 4.6%

- The report states that both models were pretrained on more than 32 trillion tokens.

Source: [DeepSeek-V4 technical report](#)

DeepSeek-V4: total capacity versus active compute (2)

- All parameters require storage; the router activates a small expert subset for each token.
- Active parameters approximate per-token model computation, not the number of stored facts.
- **Conclusion:** the architecture separates total parameter storage from the parameters used to process each token.

Source: [DeepSeek-V4 technical report](#)

Roadmap

1. Completion → inference
2. Factual knowledge
3. **Reasoning evidence**
4. Exact tools

Chain-of-thought adds test-time computation

System:

Before answering:

1. Identify the person's birth and death years.
2. Compare the requested year with that interval.
3. Return the conclusion after the intermediate steps.

- Instructions or worked examples can elicit intermediate tokens before the answer.
- Reasoning post-training can make longer intermediate generation the default behavior.
- Each added token supplies another serial model step at inference time.

The trace can improve task accuracy without becoming a formally checked proof.

Sources: [Wei et al. \(2022\)](#) · [DeepSeek-R1](#)

Training reasoning with demonstrations and verifiable rewards

Stage	Training or inference signal
Supervised fine-tuning	imitate worked solutions and answer formats
Reinforcement learning	reward answers that an exact checker accepts
Test-time scaling	sample, revise, or search through more candidate solutions
Verification	select a candidate by an outcome or process score

problem → model rollouts → verifier → reward or selection

Training changes which reasoning behavior the model generates. Test-time scaling spends more computation on a particular problem.

Experiments: when reasoning traces fail

- **Bias-cue experiment (Turpin et al.):** add a controlled cue; compare the answer and explanation.
 - Answers follow the cue, but explanations often omit it: the trace may be unfaithful.
- **Complexity experiment (Dziri et al.):** increase the depth and width of composed computations.
 - Errors rise and propagate: a fluent trace may be logically invalid.

Evaluate **answer correctness**, **stepwise logical validity**, and **causal faithfulness** separately.

Conclusion: a readable explanation can pass one test and fail the others.

Sources: [Turpin et al. \(2023\)](#) · [Dziri et al. \(2023\)](#)

Benchmarks: fact recall, inference, and faithfulness

Target	Examples	Required control
Fact recall	LAMA	paraphrases, entity frequency, cutoff date
Mathematical inference	GSM8K	answer checking, contamination, compute budget
Compositional inference	multiplication; logic-grid puzzles	computation depth and novel structures
Trace faithfulness	controlled prompt-cue interventions	compare answer changes with stated reasons

Report final-answer accuracy separately from trace validity and faithfulness. Include malformed outputs and abstentions in the scoring rule.

Roadmap

1. Completion → inference
2. Factual knowledge
3. Reasoning evidence
4. **Exact tools**

Exact tools: calculate, infer, and prove

Tool	Exact operation
Calculator or code	evaluate specified arithmetic
Rule engine	derive consequences of supplied facts and rules
Mathematical solver	solve a formal equation or constraint set
Proof checker	accept or reject a proof in a stated logic

```
FORMAL INPUT                                     [dates encoded as YYYYMMDD integers]
born(da, 19520311)
died(da, 20010511)
rule: born(P,B) and died(P,D) and B <= T < D -> alive_on(P,T)
goal: alive_on(da, 19800101)

PROVER RESULT: VALID
```

The prover checks the formal input. A wrong entity, date encoding, or rule can still

Review: was Douglas Adams alive in 1980?

```
retrieved facts: born 1952-03-11; died 2001-05-11
supplied rule:   alive_on(P,T) if born(P,B) and died(P,D)
                  and B <= T < D
witness in 1980: T = 1980-01-01
checked result:  1952-03-11 <= 1980-01-01 < 2001-05-11

answer: Yes
```

- Completion can emit a fact; retrieval supplies premises; logical inference derives a new conclusion.
- Behavioral tests measure access. Probes and interventions support narrower claims about mechanism.
- Storage and exposure estimates need a stated data model, training regime, and error definition.

- Chain-of-thought is generated computation, not a guaranteed proof.

Conclusions

- Next-token prediction can support factual recall and multi-step inference, but its behavior depends on the prompt and training distribution.
- Factual access, storage capacity, and representation inside the model require different experiments.
- Generated reasoning must be evaluated separately for answer correctness, logical validity, and causal faithfulness.
- Exact tools can check formal operations, but they cannot correct a mistaken translation from language into formal premises.

Final review

- What distinguishes next-token completion, factual retrieval, and logical inference?
- What do behavioral tests, capacity measurements, and causal interventions each establish?
- How do answer correctness, logical validity, and causal faithfulness differ?
- Which operations can exact tools verify, and which errors can still enter during formalization?

Augmented Language Models

Frameworks, knowledge access, and executable skills

Thomas M. Breuel

ICDAR 2026 Tutorial

Objectives

- Distinguish framework, knowledge-access, and skill augmentations.
- Explain the roles of RAG, GraphRAG, retrieval tools, and embeddings.
- Choose augmentations for a complex task and verify the resulting answer.

Example

Why did XYZ Corporation's diluted earnings per share change this quarter?

	2026 Q2	2025 Q2
Net income	\$312 M	\$241 M
Diluted shares	168 M	170 M
Diluted earnings/share	\$1.86	\$1.42

The answer needs facts from the report, a calculation, and evidence for each input.

question + knowledge + executable skills + control -> supported answer

Roadmap

1. **Frameworks — organize model calls and actions**
2. Information retrieval tools — add knowledge and evidence
3. Thinking tools — add executable skills

Prompt augmentation (RAG and GraphRAG)

RAG: question -> retrieve text passages -----> prompt -> LLM

GraphRAG: question -> find graph seeds -> follow graph
-> text/summaries -> prompt -> LLM

- Both methods put retrieved material in the prompt before generation.
- GraphRAG uses graph structure to select or aggregate the material.
- A retrieval tool differs because the model can decide when and what to search.

For the EPS question, the prompt can include the EPS table and the diluted-share note.

Sources: [RAG](#) · [GraphRAG](#) · [GraphRAG survey](#)

Where embeddings enter

Method	Embeddings can rank
RAG	query against text passages
GraphRAG	query against seed entities, text units, or summaries
document-search tool	query against document paragraphs
knowledge-base tool	phrase against entity names or descriptions

Exact graph traversal and exact claim lookup do not require embeddings.

For the EPS question, an embedding can rank the passage that contains “diluted earnings per share.” A similarity score ranks candidates; it does not verify a fact.

Sources: [RAG survey](#) · [GraphRAG survey](#)

Chain-of-thought

A prompt can require an intermediate reasoning trace that is separate from the final answer:

System:

Before answering, work through these steps:

1. Extract net income and diluted shares for both periods.
 2. Recompute earnings per share.
 3. Compare the income and share-count effects.
 4. Check each input against its cited passage.
- Return the conclusion separately from the reasoning trace.

Chain-of-thought prompting elicits this behavior through instructions or demonstrations. Reasoning models can also be trained and reinforced to produce extended intermediate reasoning. Both operate within a model invocation; neither supplies external evidence or an executable check.

Looping

```
framework:  
input -> GENERATE / REFINE call -> candidate -> EVALUATE call -> pass -> answer  
      ^                                     |  
      +--- input + candidate + feedback <--+--- fail
```

- Generation/refinement and evaluation are separate model invocations.
- They may use the same underlying model or different specialist models.
- The framework carries the input, candidate, and feedback and enforces the retry limit and stopping rule.

For the EPS question, the evaluation call can check the extracted values, calculation, and citations before the framework accepts the answer.

The loop is external framework control, not chain-of-thought inside one generation.

Structured call graphs

A developer usually constructs the graph and assigns each subtask to a specialist prompt or model:

```
question -> coordinator
               +-> extraction model: reported values --+
               +-> retrieval model: source passages ---+-> verifier -> answer
               +-> code model: calculate change -----+
```

The framework passes defined outputs along the edges and can run independent nodes in parallel.

A model can also propose or optimize a call graph. The framework still executes candidate graphs and evaluates them on specified examples and metrics.

Sources: [DSPy](#) · [Automated Design of Agentic Systems](#)

Tool use

An MCP server publishes a tool name, description, and parameter schema. A small `mcp-this` configuration can expose an existing command-line program:

```
tools:
  pdf-to-text:
    description: Extract layout-preserving text from a PDF file.
    execution:
      command: "pdftotext -layout <<path>> -"
    parameters:
      path:
        description: Path to the PDF file
        required: true
```

The MCP client can now call `pdf-to-text(path="quarterly-report.pdf")`. The host must authorize the call and validate or sandbox command arguments.

Self-improvement

- prompt tuning: instructions, demonstrations, and tool descriptions;
- framework-controlled generation, evaluation, and refinement loops;
- call-graph generation, search, and optimization;
- retrieval-source discovery and index tuning;
- training-data selection, filtering, and augmentation for pretraining and supervised fine-tuning;
- preference-data generation, reward modeling, and reinforcement learning;
- generation of reusable skills and procedures.

For the EPS task, checked filing questions can score candidate extraction prompts and call graphs.

Scratchpad memory

An agent workspace can contain:

- downloaded documents, web pages, and images;
- OCR output, extracted tables, and document fragments;
- generated summaries, outlines, and source notes;
- intermediate calculations, checklists, and results stored as Markdown or structured files;
- state retained across model invocations and sessions;
- automatically generated skills or instructions that record a reusable procedure.

These files are external working memory. Their provenance, version, and validity still need to be tracked.

Pi Agent Harness

- Pi selects a provider and model, then sends a system prompt, project context files, session messages, and tool schemas.
- The model can request the default `read`, `write`, `edit`, and `bash` tools or tools added by extensions.
- Pi executes the calls, appends the tool results, and invokes the model again until it returns a final response or the runtime stops the loop.
- Sessions are stored as JSONL trees; context compaction summarizes older messages before another model call.

Source: [Pi coding agent](#)

Hermes Agent Runtime

- Hermes assembles a cached system prompt from identity and tool guidance, skills, project context, memory, and the user profile.
- A provider resolver maps the selected provider and model to its API format before each model invocation.
- If the model returns tool calls, Hermes dispatches them through its tool registry, appends the results, and invokes the model again.
- The loop handles compression, retries, model fallback, and session persistence; SQLite and FTS5 support session recall.

Source: [Hermes architecture](#)

OpenClaw Agent Runtime

- OpenClaw resolves the session, workspace, model provider, and agent runtime for each incoming message.
- It assembles the system prompt from base instructions, skills, workspace files, memory, session history, and available tool schemas.
- A serialized per-session loop invokes the model, executes policy-approved tool calls, appends results, and invokes the model again.
- The runtime streams the final response to the originating channel and persists the transcript; sandboxing constrains tool execution.

Source: [OpenClaw agent runtime](#)

DeepSeek Harness and Cordis

```
Cordis kernel -----+--> model plugin
                      +--> tool and skill plugins
                      +--> session and storage plugins
                      +--> loop and scheduling plugins
                      +--> sandbox and interface plugins
```

- Plugins declare services and dependencies instead of relying on a fixed load order.
- Reversible effects support mounting, replacing, and removing components.
- Configuration selects the agent composition without changing the harness source.

DeepSeek Harness is a developer preview. The Cordis paper explains its composition model, not task-performance gains.

Sources: [DeepSeek Harness](#) · [Cordis paper](#)

Roadmap

1. Frameworks — organize model calls and actions
2. **Information retrieval tools — add knowledge and evidence**
3. Thinking tools — add executable skills

LLM delegation

```
EPS question -> lead +-> document analyst: find values and passages
+-> accounting analyst: explain the calculation
+-> reviewer: reject unsupported claims
```

The lead model compares the returned records before it writes the answer.

- Advantage: more computation, specialization, and independent task decomposition.
- Limitation: another model response is generated analysis, not primary evidence.
- Cost: extra calls add latency, and shared training can produce correlated errors.

Sources: [AutoGen](#) · [Multiagent debate](#) · [Mixture-of-Agents](#)

Web search

```
search("site:sec.gov XYZ Corporation 2026 Q2 10-Q")  
-> inspect results  
-> open the official filing  
-> locate the EPS and diluted-share notes  
-> retain URL, filing date, page, and passage
```

The result list supplies candidates. The opened filing supplies evidence.

If the report is missing, the agent can reformulate the query or report failure rather than inventing a value.

Sources: [WebGPT](#) · [Search-R1](#)

Document search

Build a fixed semantic index from a dated Wikimedia dump:

```
articles -> paragraphs -> embeddings -> FAISS index  
  
search_paragraphs("British science-fiction author")  
-> [(0.82, page="Douglas Adams", paragraph=1,  
      text="English author and screenwriter ..."), ...]
```

The score is illustrative. A result contains the matched text and its page metadata; the score measures similarity, not truth.

For the EPS question, the same interface can search paragraphs extracted from the quarterly report.

Sources: [RAG](#) · [Atlas](#)

Knowledge-base interaction

Wikidata tools return identifiers and structured claims:

```
name_to_id("Douglas Adams")      -> [Q42, ...]
id_to_name(Q42)                  -> "Douglas Adams"
values(Q42, P19)                  -> [Q350]
id_to_name(Q350)                  -> "Cambridge"
subjects(P31, Q5)                 -> [Q42, ...]
semantic_entity("British sci-fi author") -> [(0.79, Q42), ...]
```

Exact lookups use structured indexes; semantic name and description lookup uses FAISS to propose candidates. For the EPS task, a general knowledge base can identify the company and its relationships, while the quarterly values remain in the filing.

Sources: [Wikidata](#) · [SPINACH](#)

Roadmap

1. Frameworks — organize model calls and actions
2. Information retrieval tools — add knowledge and evidence
3. **Thinking tools — add executable skills**

Information extraction

- OCR converts document images into text, layout, and table cells.
- Vision-language models recognize visual entities, figures, page regions, and their relationships.
- Language models extract named entities, relations, events, quantities, dates, and qualifications from text.
- Parsers and validators convert model output into a defined schema and reject malformed records.
- Entity linking maps extracted names and relations to stable identifiers.

For the EPS question, extraction produces the company, metric, period, value, unit, and source location.

Procedural tools

Tool	Defined operation
calculator	exact arithmetic
syntax checker	accept or reject a formal input
test runner	execute stated tests and report failures
code executor	run a bounded program and return its output

For the EPS question, a calculator makes the central result reproducible:

$$(312 / 168) / (241 / 170) - 1 = 0.310 \rightarrow 31.0\%$$

The model still must choose the correct inputs and operation; code execution also needs a sandbox.

Reasoning tools

Tool	Result
expert system	a conclusion with a rule trace
inference engine	consequences of facts and rules
mathematical solver	a symbolic or numeric solution
proof checker	acceptance or rejection of a proposed proof

```
IF abs(income_effect) > abs(share_count_effect)
THEN primary_driver = net_income
```

A reasoning tool guarantees only what follows from its supplied formalization. The model can still translate the question, facts, or rules incorrectly.

Return to the example

reported facts + calculation + rule -> supported explanation

XYZ Corporation's diluted EPS increased from \$1.42 to \$1.86, or 31.0%. Higher net income accounts for most of the increase; the lower diluted-share count contributes less.

Evidence returned with the answer:

- reported values and net income: quarterly report, p. 34;
- diluted share counts: quarterly report, p. 41;
- percentage and attribution: stored calculation and rule trace.

The report values are fictional. The method separates reported facts from derived conclusions.

Review

You should now be able to:

- distinguish frameworks, knowledge access, and executable skills;
- explain how RAG, GraphRAG, and model-invoked retrieval differ;
- state where embeddings rank candidates and where exact lookup takes over;
- choose among delegation, web search, document search, and knowledge-base interaction;
- use calculators, code, rules, and provers for defined operations while checking the model's translation.