Intent Modeling Language for Text-to-SQL Queries

Precise and Deterministic Ambiguity Management for Human-AI Misalignment Risks in Text-to-SQL

Ron Itelman
ron@ronitelman.com

July 2025

Abstract

This paper introduces an Intent Modeling Language (IML), a deterministic framework for precisely and deterministically identifying, quantifying, and resolving ambiguity in natural language to SQL systems **before** engaging large language models. This paper offers a clear paradigm shift from reactive to proactive disambiguation, positioning Intent Modeling as a deterministic layer that precedes LLM engagement. It's especially impactful for industries requiring high reliability (e.g., finance, healthcare, law).

1 Introduction to Ambiguity in Natural Language Queries

Let's examine a global financial trading environment, and how a simple query like "Show my trades from yesterday" creates a cascade of temporal interpretation challenges:

Example natural language query problem: "Show my trades from yesterday"

When a UK trader in London queries a business platform hosted in New York about trades executed in Sydney, the word "yesterday" can legitimately refer to three different time zones, each representing a distinct stakeholder perspective. This temporal ambiguity represents a broader class of human-AI misalignment risks that occur when natural language interfaces must bridge multiple conceptual frameworks.

1.1 Key Terms

To establish clarity throughout this paper, we define the following key terms and their relationships contained within Intent Modeling Language (IML):

Intent Modeling: The core methodology for unifying meaning in a general way, across systems that is both human and machine readable. Intent Modeling creates shareable, transformable units of logic that preserve intent across fragmented organizational systems.

Intent Matrix: The specific implementation tool within Intent Modeling that provides quantifiable ambiguity scoring mechanisms for systematic measurement and resolution of ambiguity through deterministic calculations rather than probabilistic approaches.

Intent Graph: The workflow and decision representations within Intent Modeling that structures sequential processing logic for multi-step ambiguity resolution.

Intent Engineering: The disciplined practice of systematically managing intent and ambiguity resolution within organizational systems. Intent Engineering transforms ambiguity management from an ad-hoc art into a measurable, predictable engineering discipline with systematic assessment criteria and improvement pathways.

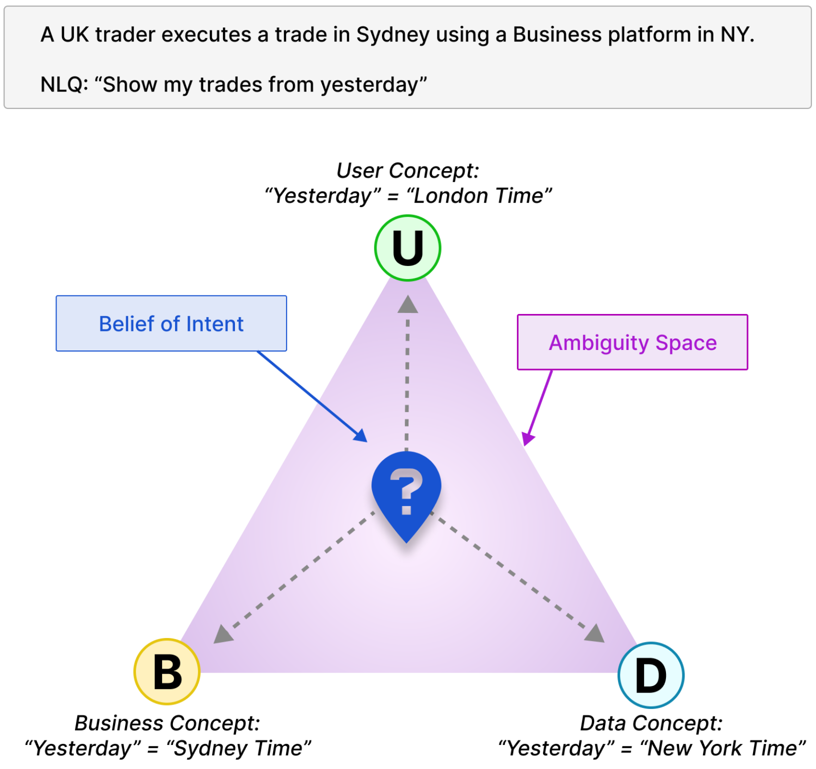
1.2 Novelty to the Semantic and AI Industry

At the time of this writing, no other known methodology exists to identify, quantify, and mitigate ambiguity in a formal engineering discipline.

2 Intent Modeling Language

Intent Modeling Language is an umbrella concept that integrates three core components to create systematic ambiguity management:

1. **Ambiguity Space Model**: Identifies specific contexts where temporal references create interpretation conflicts. This simply shows how many possible meanings can occur. For our example of "yesterday", we have three options: From the perspective of the trader, the exchange, or the system the trader uses.


A UK trader executes a trade in Sydney using a Business platform in NY.

NLQ: “Show my trades from yesterday”
User Concept:
“Yesterday” = “London Time”
U
Belief of Intent
Ambiguity Space
?
B
Business Concept:
“Yesterday” = “Sydney Time”
D
Data Concept:
“Yesterday” = “New York Time”

Figure 1: *There are multiple possible directions to assign our belief of the user’s intent that are all equally viable.*

2. **Ambiguity Layer Model**: A tree of intervention points across the technology stack. We start primarily with four layers, and each layer can have sub-layers.

   - *Logic*: Ambiguity in the business logic and/or governance that other teams are to be aligning to.

- *UX*: Where the user is entering their natural language query, getting responses from the LLM, and receiving UI-based clarifying questions.
- *Context Engineering*: Typically for context/prompt engineering scripts and processes that are added to a message going to an LLM that the user doesn't see on the front-end (UX layer). Example sub-layers:
  - *NLQ-to-SQL ambiguity*: unclear or undocumented conversion from natural language queries to SQL in the prompt engineering script sent in the message
- *Data*: The actual SQL, dataset, or code processing used internally by a system for SQL/API/JSON translation. Example sub-layers:
  - *Column ambiguity*: unclear or undocumented column names and table structures.

Context Engineering Space: Layer Examples

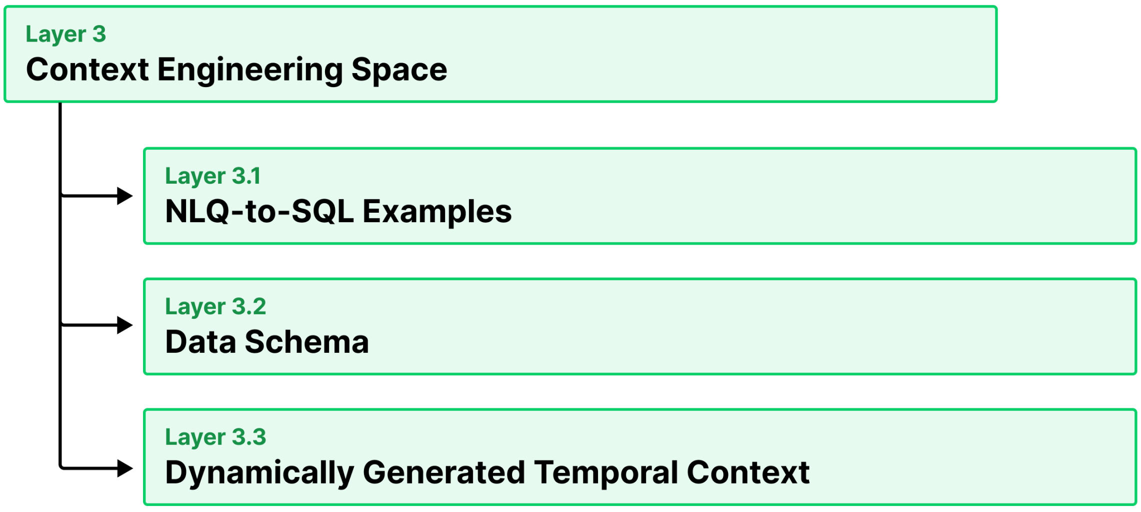


Figure 2: *An Ambiguity Space Layer may have sub-layers*

3. **Ambiguity Minimization Model**: By identifying precisely where ambiguity exists using the Ambiguity Space and Ambiguity Layer models, we can ask clarifying questions as feedback loops. In order to minimize ambiguity, we first must be able to measure it.

   Our Intent Modeling Language uses the following definition: of **ambiguity score = 1 - unresolved slots / total slots** This primarily comes from a combination of the following:

   The `Intent Ambiguity Score` is the inverse of unresolved/resolved slots. So if there are 5 slots: `(1 - (resolved = 0 / total-to-resolve = 5))` → becomes 1 - 0, or 100% ambiguity. If all 5 slots are resolved 1-1=0, so 0% ambiguity.

Listing 1: Intent Map Example with Ambiguity Scoring

```
{
    "intent-map-id": 1,
    "intent-command": "/concept-registry/intent/command/get-trades-by-date-range",
    "intent-schema": "CREATE TABLE trades (
                    trade_id INTEGER PRIMARY KEY AUTOINCREMENT,
                    trader_id TEXT NOT NULL,
                    trade_date DATE NOT NULL,
                    bond_symbol TEXT NOT NULL,
                    side TEXT NOT NULL,
                    quantity REAL NOT NULL,
                    price REAL NOT NULL
                );",
    "intent-query": "SELECT * FROM trades WHERE
                    trader_id = :trader_id AND
                    trade_date BETWEEN :start_date AND end_date;",
    "intent-sql-slots": [
        "/concept-registry/intent/slots/sql/temporal/start_date",
        "/concept-registry/intent/slots/sql/temporal/end_date"
    ],
    "intent-nlq-slots": [
        "/concept-registry/intent/slots/nlq/temporal/perspective-trader",
        "/concept-registry/intent/slots/nlq/temporal/perspective-exchange",
        "/concept-registry/intent/slots/nlq/temporal/perspective-system"
    ],
    "intent-sql-slots-state-resolved": [],
    "intent-nlq-slots-state-resolved": [],
    "intent-sql-slots-state-to_resolve-count": 2,
    "intent-nlq-slots-state-to_resolve-count": 3,
    "intent-sql-slots-state-is_resolved-count": 0,
    "intent-nlq-slots-state-is_resolved-count": 0,
    "intent-map-ambiguity-score": 1.0
}
```

The framework enables organizations to move beyond ad-hoc solutions toward systematic measurement and control of temporal ambiguity risks in human-AI systems.

3 Defining a "Concept"

Semantics often uses the term "entity" to refer to a "thing" such as "trades" in a natural language query.

For Intent Modeling, a *Concept* is the human and machine readable representation of precisely measuring where ambiguity exists and how to resolve it:

Listing 2: Create trades table

```
CREATE TABLE trades (
    trade_id INTEGER PRIMARY KEY AUTOINCREMENT,
    trader_id TEXT NOT NULL,
    trade_date DATE NOT NULL,
    bond_symbol TEXT NOT NULL,
    side TEXT NOT NULL,
    quantity REAL NOT NULL,
    price REAL NOT NULL
);
```

There are a few key pieces of information missing:

- *Multiple perspectives of trade date*: We don't capture that there are multiple perspectives of *trade date*: The trader, the exchange, or the system. This logic is not typically captured and is critical for the act of *how to engineer an intent*, because an intent map captures a table of ambiguity and how to resolve it.

Each concept is a URI that can link to other concepts and is viewable in a concept-registry: "/concept-registry/intent/command/get-trades-by-date-range"

Listing 3: Concept JSON Example

```
{
  "concept-id": "/concept-registry/intent/command/get-trades-by-date-range",
  "layer": "intent",
  "type": "command",
  "index": "get-trades-by-date-range",
  "slots": {
    "sql": [
      "/concept-registry/intent/parameter/date-range-start",
      "/concept-registry/intent/parameter/date-range-end"
    ],
    "nlq": [
      "/concept-registry/intent/parameter/temporal-perspective-trader",
      "/concept-registry/intent/parameter/temporal-perspective-exchange",
      "/concept-registry/intent/parameter/temporal-perspective-system"
    ]
  },
  "description": "This intent command is for getting trades by a date range",
  "aliases": ["orders", "transactions", "positions", "trades"],
  "nlq-negative-examples": [
    "place an order",
    "cancel my trade",
    "update order",
    "create new position",
    "set stop loss",
    "modify trade"
  ],
  "nlq-positive-examples": [
        "show my orders",
        "show order",
        "show orders",
        "get my orders",
        "get orders",
        "get my trades",
        "get trades",
        "list my orders"
      ]
}
```

4 Intent Matrix Ambiguity Score Calculation

4.1 Overview

The Intent Matrix Ambiguity Score combines two dimensions of ambiguity in natural language query processing:

1. **Command Ambiguity**: Uncertainty about which intent/command the user intends
2. **Slot Ambiguity**: Uncertainty about the parameter values needed to execute the resolved intent

4.2 Formula

$$\text{intent-matrix-ambiguity-score} = \text{command\_ambiguity} + (1 - \text{command\_ambiguity}) \times \text{average\_slot\_ambiguity} \tag{1}$$

Where:

$$\text{command\_ambiguity} = 1 - \frac{\text{resolved\_commands}}{\text{total\_commands}} \tag{2}$$

$$\text{average\_slot\_ambiguity} = \text{mean of all intent-map-ambiguity-scores in the matrix} \tag{3}$$

4.3 Step-by-Step Calculation

4.3.1 Step 1: Calculate Command Ambiguity

$$\text{command\_ambiguity} = 1 - \frac{\text{intent-matrix-command-slots-is\_resolved-count}}{\text{intent-matrix-command-slots-to\_resolve-count}} \tag{4}$$

Example:

- `intent-matrix-command-slots-is_resolved-count`: 0
- `intent-matrix-command-slots-to_resolve-count`: 2
- $\text{command\_ambiguity} = 1 - \frac{0}{2} = 1.0$ (100% ambiguous)

4.3.2 Step 2: Calculate Average Slot Ambiguity

Sum all individual `intent-map-ambiguity-score` values and divide by the number of intent maps:

$$\text{average\_slot\_ambiguity} = \frac{\sum(\text{intent-map-ambiguity-score})}{\text{number\_of\_intent\_maps}} \tag{5}$$

Example:

- Intent Map 1: `intent-map-ambiguity-score` $= 1.0$
- Intent Map 2: `intent-map-ambiguity-score` $= 1.0$
- average_slot_ambiguity $= \frac{1.0+1.0}{2} = 1.0$

4.3.3 Step 3: Apply Hierarchical Formula

$$\text{intent-matrix-ambiguity-score} = \text{command\_ambiguity} + (1 - \text{command\_ambiguity}) \times \text{average\_slot\_ambiguity} \tag{6}$$

Example:

$$\begin{aligned} \text{intent-matrix-ambiguity-score} &= 1.0 + (1 - 1.0) \times 1.0 && (7) \\ &= 1.0 + 0.0 \times 1.0 && (8) \\ &= 1.0 && (9) \end{aligned}$$

4.4 Interpretation

- **Score = 1.0**: Maximum ambiguity (100%)
- **Score = 0.0**: No ambiguity (0%)
- **Score between 0-1**: Partial ambiguity

The hierarchical nature means:

- If command ambiguity is high, overall ambiguity will be high regardless of slot resolution
- As command ambiguity decreases, slot-level ambiguity becomes the limiting factor
- Only when both dimensions are resolved does overall ambiguity approach zero

4.5 Example Scenarios

4.5.1 Scenario 1: Initial State (Complete Ambiguity)

- Commands resolved: $0/2 \rightarrow$ `command_ambiguity` $= 1.0$
- Average slot ambiguity: 1.0
- **Result**: $1.0 + (1 - 1.0) \times 1.0 = 1.0$

4.5.2 Scenario 2: Command Resolved, Slots Partially Resolved

- Commands resolved: $1/2 \rightarrow$ `command_ambiguity` $= 0.5$
- Average slot ambiguity: 0.6 (some slots filled)
- **Result**: $0.5 + (1 - 0.5) \times 0.6 = 0.5 + 0.3 = 0.8$

4.5.3 Scenario 3: Command Resolved, All Slots Resolved

- Commands resolved: $1/2 \rightarrow$ `command_ambiguity` $= 0.5$
- Average slot ambiguity: 0.0 (all slots filled)
- **Result**: $0.5 + (1 - 0.5) \times 0.0 = 0.5$

4.5.4 Scenario 4: Complete Resolution

- Commands resolved: $1/1 \rightarrow$ `command_ambiguity` $= 0.0$
- Average slot ambiguity: 0.0
- **Result**: $0.0 + (1 - 0.0) \times 0.0 = 0.0$

4.6 Implementation Notes

- The formula naturally prioritizes command resolution before slot resolution
- Edge case: When `intent-matrix-command-slots-to_resolve-count` $= 1$, command ambiguity becomes binary (1.0 or 0.0)
- Consider implementing bounds checking to ensure the score remains within [0, 1]
- The calculation should be performed each time slot states change in any intent map

4.7 Intent Matrix JSON Example

Listing 4: Intent Matrix JSON Example

```
{
  "intent-maps": [
    {
      "intent-map-id": 1,
      "intent-command": "/concept-registry/intent/command/get-trades-by-date-range",
      "intent-schema": "CREATE TABLE trades (trade_id INTEGER PRIMARY KEY AUTOINCREMENT, trader_id TEXT NOT NULL, trade_date DATE NOT NULL, bond_symbol TEXT NOT NULL, side TEXT NOT NULL, quantity REAL NOT NULL, price REAL NOT NULL);",
      "intent-query": "SELECT * FROM trades WHERE trader_id = :trader_id AND trade_date BETWEEN :start_date AND end_date;",
      "intent-sql-slots": ["/concept-registry/intent/slots/sql/temporal/start_date", "/concept-registry/intent/slots/sql/temporal/end_date"],
      "intent-nlq-slots": ["/concept-registry/intent/slots/nlq/temporal/perspective-trader", "/concept-registry/intent/slots/nlq/temporal/perspective-exchange", "/concept-registry/intent/slots/nlq/temporal/perspective-system"],
      "intent-sql-slots-state-resolved": [],
      "intent-nlq-slots-state-resolved": [],
      "intent-sql-slots-state-to_resolve-count": 2,
      "intent-nlq-slots-state-to_resolve-count": 3,
      "intent-sql-slots-state-is_resolved-count": 0,
      "intent-nlq-slots-state-is_resolved-count": 0,
      "intent-map-ambiguity-score": 1.0
    },
    {
      "intent-map-id": 2,
      "intent-command": "/concept-registry/intent/command/get-trades-by-date-range",
      "intent-schema": "CREATE TABLE trades (trade_id INTEGER PRIMARY KEY AUTOINCREMENT, trader_id TEXT NOT NULL, trade_date DATE NOT NULL, bond_symbol TEXT NOT NULL, side TEXT NOT NULL, quantity REAL NOT NULL, price REAL NOT NULL);",
      "intent-query": "SELECT * FROM trades WHERE ...",
      "intent-sql-slots": ["/concept-registry/intent/slots/sql/temporal/start_date", "/concept-registry/intent/slots/sql/temporal/end_date"],
      "intent-nlq-slots": ["/concept-registry/intent/slots/nlq/temporal/perspective-trader", "/concept-registry/intent/slots/nlq/temporal/perspective-exchange", "/concept-registry/intent/slots/nlq/temporal/perspective-system"],
      "intent-sql-slots-state-resolved": [],
      "intent-nlq-slots-state-resolved": [],
      "intent-sql-slots-state-to_resolve-count": 2,
      "intent-nlq-slots-state-to_resolve-count": 3,
      "intent-sql-slots-state-is_resolved-count": 0,
      "intent-nlq-slots-state-is_resolved-count": 0,
      "intent-map-ambiguity-score": 1.0
    }
  ],
    "intent-matrix-command-slots": ["/concept-registry/intent/command/get-trades-by-date-range", "/concept-
     registry/intent/command/get-trades-by-date"],
    "intent-matrix-command-slots-is_resolved-count": 0,
    "intent-matrix-command-slots-to_resolve-count": 2,
    "intent-matrix-command-ambiguity-score": 1.0,
    "intent-matrix-total-ambiguity-score": 1.0
}
```

5 Intent Graph

Just as a Directed Acyclic Graph can map out sequential steps, when we need to process the sequential steps of Intent Matrix Ambiguity Scores, or Intent Matrix Processing Logic, an *Intent Graph* is merely an array of intent processing / reasoning logic. However, Intent Graphs are beyond the scope of this paper.

6 Semantic Threat Example: No Way to Know Intent Despite "Perfect" NLQ-to-SQL Conversion

6.1 SQL Query Design for Bond Trading System

6.1.1 Database Schema

The following table structure captures the multi-timezone complexity inherent in global bond trading:

```
CREATE TABLE bond_trades (
    trader_id VARCHAR(50) NOT NULL,
    exchange_name VARCHAR(100) NOT NULL,
    trader_location VARCHAR(100) NOT NULL,
    trade_timestamp_system TIMESTAMP NOT NULL,    -- NY system time
    trade_timestamp_exchange TIMESTAMP NOT NULL,  -- Exchange local time
    trade_timestamp_trader TIMESTAMP NOT NULL,    -- Trader local time
    trade_amount DECIMAL(15,2),
    bond_symbol VARCHAR(20),
    PRIMARY KEY (trader_id, trade_timestamp_system)
);
```

This schema is designed to explicitly capture the temporal ambiguity that exists in global trading systems. By storing three separate timestamps, we acknowledge that a single "trade event" exists simultaneously in multiple temporal contexts, each potentially relevant for different business purposes, regulatory requirements, and operational decisions.

6.1.2 Perspective-Based Query Examples

Each stakeholder's definition of "yesterday's trades" reflects their operational context:

System Perspective Query:

```
SELECT * FROM bond_trades
WHERE DATE(trade_timestamp_system) = CURRENT_DATE - INTERVAL '1' DAY
AND trade_timestamp_system >= '2024-01-15 00:00:00 EST'
AND trade_timestamp_system < '2024-01-16 00:00:00 EST';
```

This query prioritizes system operational consistency. It uses the system's authoritative timestamp (NY time) to define the trading day, ensuring consistent reporting and settlement processing regardless of where trades originated globally.

Exchange Perspective Query:

```
SELECT * FROM bond_trades
WHERE DATE(trade_timestamp_exchange) = CURRENT_DATE - INTERVAL '1' DAY
AND exchange_name = 'Australia'
AND trade_timestamp_exchange >= '2024-01-15 00:00:00 AEST'
AND trade_timestamp_exchange < '2024-01-16 00:00:00 AEST';
```

This query reflects regulatory and market-specific requirements. Exchanges must report according to their local trading sessions and regulatory frameworks, making the exchange timestamp the authoritative source for compliance and market analysis.

Trader Perspective Query:

```
SELECT * FROM bond_trades
WHERE DATE(trade_timestamp_trader) = CURRENT_DATE - INTERVAL '1' DAY
AND trader_id = 'TR001'
AND trade_timestamp_trader >= '2024-01-15 00:00:00 PST'
AND trade_timestamp_trader < '2024-01-16 00:00:00 PST';
```

This query serves individual performance evaluation and workflow management. Traders operate within their local context, and their "trading day" aligns with their personal schedule and local market hours, affecting performance metrics and workload distribution.

6.2 Quantifying and Locating Intent Ambiguity

Despite perfect NLQ to SQL conversion, without mapping exactly which perspective is being used, there's no way to know which NLQ to SQL conversion to use!

Take for example this table, where we have a single table schema, and a single query we want to convert from NLQ to SQL. Only one of these rows is true, but without a logical structure to map this together any one of these are equally possibly true.

TABLE ID	**QUERY ID**	**TRADER PERSPECTIVE**	**SYSTEM PERSPECTIVE**	**EXCHANGE PERSPECTIVE**
142	1	✓	X	X
142	1	X	✓	X
142	1	X	X	✓

Table 1: Intent Ambiguity in SQL Query Interpretation

This table demonstrates a fundamental challenge in data systems: even with syntactically perfect SQL queries operating on the same table schema, the *intent* behind each query remains ambiguous without explicit context. Each query returns different result sets from the same underlying data, yet all three could be considered "correct" implementations of "yesterday's trades."

The value of an Intent Map is the ability to precisely quantify how much ambiguity exists (the possible matches of queries and tables), as well as how many possible slots there are.

For example, with only two perspectives, there is a 0.5 chance of guessing it right, with 10 perspectives, there is only a 0.1 chance of guessing.

7 Related Work and Literature Review

The challenge of ambiguity in natural language processing has been extensively studied across multiple domains. However, existing approaches fundamentally differ from our methodology in their focus on *resolving* ambiguity through probabilistic methods rather than *quantifying* ambiguity through deterministic measurement before machine learning engagement.

7.1 Semantic Disambiguation and Word Sense Disambiguation

Traditional semantic disambiguation research has focused primarily on word sense disambiguation (WSD) and lexical ambiguity resolution. Pal and Saha [3] provide a comprehensive survey of WSD approaches, demonstrating that most methods rely on statistical and machine learning algorithms to select the most likely meaning from a set of alternatives. More recently, Abeysiriwardana and Irugalbandara [4] explored modern deep learning techniques for lexical ambiguity detection, highlighting advances in neural approaches but noting persistent challenges with sense-annotated corpora scarcity.

Loureiro et al. [5] conducted an in-depth analysis of transformer-based language models like BERT for word sense disambiguation, showing that while these models can capture high-level sense distinctions effectively, they still operate on probabilistic confidence measures rather than deterministic ambiguity quantification. Martinez-Gil [6] proposed unsupervised approaches using context-aware semantic similarity, but again focused on disambiguation accuracy rather than systematic ambiguity measurement.

While these approaches achieve reasonable performance in disambiguation tasks, they fundamentally operate on a *confidence-based paradigm*. A system may report 85% confidence in a disambiguation decision, but this confidence score does not quantify the underlying ambiguity space or provide systematic measurement of unresolved semantic slots. The confidence metric reflects the system's certainty about its chosen interpretation, not the inherent ambiguity present in the input.

7.2 Intent Recognition and Classification Systems

Modern intent recognition systems have made significant advances in understanding user intentions from natural language input. Liu and Lane [7] introduced recurrent neural networks for joint intent detection and slot filling in spoken language understanding, demonstrating improved performance through joint training approaches. However, their evaluation metrics focused on classification accuracy rather than ambiguity quantification.

Recent work has explored intent detection with large language models. The authors of [8] examined intent detection capabilities of modern LLMs, showing that while these systems can handle broad-to-specific intent scopes, they still rely on confidence-based classification without systematic ambiguity measurement. Similarly, research on user intent recognition with ChatGPT [9] demonstrated improvements in intent classification accuracy but provided no framework for quantifying how much ambiguity existed in cases where classification failed.

These systems typically operate through supervised learning on labeled datasets, producing confidence scores for intent classifications. However, they suffer from the same fundamental limitation: confidence in classification is not equivalent to systematic ambiguity measurement. A system achieving 90% accuracy on intent classification provides no mechanism for quantifying how much ambiguity existed in the 10% of cases where it failed, nor does it offer structured approaches for reducing that ambiguity through deterministic intervention.

7.3 Natural Language to SQL Translation

The specific challenge of converting natural language queries to SQL has received considerable attention in recent years. Kumar et al. [10] provided a comprehensive survey of deep learning approaches for text-to-SQL conversion, reviewing 24 neural network models including convolutional neural networks, recurrent neural networks, and reinforcement learning approaches. However, their analysis focused on translation accuracy rather than ambiguity management.

Mohammadjafari et al. [11] reviewed LLM-based text-to-SQL systems, demonstrating advances in retrieval-augmented generation approaches. Wong et al. [12] achieved 73-84% exact match accuracy using T5 models, while Yang et al. [13] explored automated repair mechanisms for incorrect SQL generation.

Despite these advances, current approaches remain fundamentally probabilistic. The BIRD benchmark [1] demonstrates that even state-of-the-art systems achieve only 77.53% accuracy, compared to 92.96% human performance. More critically, these systems provide no systematic method for identifying, quantifying, or managing the sources of ambiguity that lead to translation failures.

7.4 Gap Analysis and Contribution Positioning

Our analysis reveals a fundamental gap in existing literature: while substantial research exists on *resolving* ambiguity through improved algorithms, no known methodology provides *systematic quantification* of ambiguity before machine learning engagement. Existing approaches share several limitations:

1. **Probabilistic Dependency**: All reviewed approaches rely on machine learning models that produce confidence scores rather than deterministic ambiguity measurements
2. **Post-Hoc Resolution**: Ambiguity is addressed after it has caused system confusion, rather than being identified and resolved proactively
3. **Lack of Quantification**: No systematic method exists for measuring how much ambiguity exists in a given natural language input
4. **No Structured Management**: Existing systems provide no framework for systematically reducing ambiguity through coordinated intervention across organizational and technological layers

Intent Modeling methodology addresses these gaps by providing:

- **Deterministic Quantification**: Mathematical formulas for measuring ambiguity without probabilistic dependencies
- **Pre-ML Intervention**: Systematic identification and resolution of ambiguity before engaging large language models
- **Structured Management**: The Intent Matrix and Ambiguity Layer Model provide coordinated frameworks for ambiguity management across organizational systems
- **Measurable Improvement**: Quantifiable ambiguity scores enable systematic assessment and optimization of disambiguation effectiveness

This represents a paradigm shift from *managing uncertainty in ML outputs* to *eliminating uncertainty in ML inputs*, providing organizations with deterministic control over human-AI alignment in natural language interfaces.

8 Conclusion

Intent Modeling provides a comprehensive framework for understanding and managing temporal ambiguity in natural language to SQL systems through systematic pre-LLM intervention. By integrating the Ambiguity Space Model, Ambiguity Layer Model, and Ambiguity Minimization Model, organizations gain deterministic tools for identifying risks, implementing controls, and measuring improvements without relying on probabilistic ML approaches.

The Core Insight: Intent Modeling creates a **disambiguation layer** that sits between natural language and SQL, making the system deterministic rather than probabilistic. This approach is particularly valuable for high-stakes applications where the current "77.53% accuracy ceiling" of leading AI systems (compared to 92.96% human performance) is insufficient for production deployment.

Please note: https://bird-bench.github.io/ is the reference for these statistics as of July 29, 2025

Superior Architecture Through Pre-LLM Resolution: The strategy of **avoiding ML entirely** and resolving ambiguity **before** the LLM represents a fundamental architectural advantage:

- **Deterministic vs. Probabilistic**: Organizations require predictable behavior rather than hoping improved prompting will resolve inherently ambiguous inputs
- **Systematic Resolution**: Front-end UI interventions and regex-based pattern recognition provide reliable disambiguation mechanisms
- **Measurable Outcomes**: Intent Matrix scoring enables quantifiable assessment and improvement of ambiguity management
- **Technology Stack Integration**: The Context Engineering Layer and systematic intervention points create coordinated ambiguity management across organizational layers

The framework's strength lies in its recognition that temporal ambiguity is not merely a technical problem requiring better ML models, but a systematic challenge requiring coordinated intervention across multiple organizational and technological layers. Rather than accepting the limitations of current natural language processing capabilities, Intent Modeling transforms ambiguity management from an operational risk into a systematically managed aspect of human-AI collaboration.

As AI systems become more prevalent in global business operations, deterministic frameworks like Intent Modeling become essential for maintaining human-AI alignment and ensuring reliable system behavior in enterprise environments where accuracy requirements exceed current ML capabilities.

Future Work will focus on developing automated assessment tools for each layer of the Ambiguity Layer Model, creating industry-specific Intent Matrix templates, and establishing benchmarks for measuring disambiguation effectiveness across different organizational contexts. The ultimate goal is establishing Intent Modeling as the standard approach for managing ambiguity in mission-critical human-AI systems where deterministic behavior is paramount.

References

[1] Li, J., Hui, B., Qu, G., Yang, J., Li, B., Li, B., Wang, B., Qin, B., Geng, R., Huo, N., et al. Can LLM Already Serve as A Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQL. *arXiv preprint arXiv:2305.03111*, 2023. Available at: `https://arxiv.org/pdf/2305.03111`

[2] BIRD Benchmark Contributors. BIRD: BIg Bench for LaRge-Scale Database Grounded Text-to-SQL Evaluation - Leaderboard Results. *Online Database*, March 2025. Available at: `https://bird-bench.github.io/`

[3] Pal, A. R. and Saha, D. Word sense disambiguation: a survey. *arXiv preprint arXiv:1508.01346*, 2015. Available at: `https://arxiv.org/abs/1508.01346`

[4] Abeysiriwardana, M. and Irugalbandara, I. A Survey on Lexical Ambiguity Detection and Word Sense Disambiguation. *arXiv preprint arXiv:2403.16129*, 2024. Available at: `https://arxiv.org/abs/2403.16129`

[5] Loureiro, D., Rezaee, K., Pilehvar, M. T., and Camacho-Collados, J. Analysis and Evaluation of Language Models for Word Sense Disambiguation. *arXiv preprint arXiv:2008.11608*, 2021. Available at: `https://arxiv.org/abs/2008.11608`

[6] Martinez-Gil, J. Context-Aware Semantic Similarity Measurement for Unsupervised Word Sense Disambiguation. *arXiv preprint arXiv:2305.03520*, 2023. Available at: `https://arxiv.org/abs/2305.03520`

[7] Liu, B. and Lane, I. Joint Online Spoken Language Understanding and Language Modeling with Recurrent Neural Networks. *arXiv preprint arXiv:1609.01462*, 2016. Available at: `https://arxiv.org/abs/1609.01462`

[8] Singh, P., Khandelwal, A., Agarwal, A., and Sinha, P. Intent Detection in the Age of LLMs. *arXiv preprint arXiv:2410.01627*, 2024. Available at: `https://arxiv.org/abs/2410.01627`

[9] Babaei, A., Ritter, S., and Xu, W. User Intent Recognition and Satisfaction with Large Language Models: A User Study with ChatGPT. *arXiv preprint arXiv:2402.02136*, 2024. Available at: `https://arxiv.org/abs/2402.02136`

[10] Kumar, A., Khattar, A., Vatsal, A., Singh, A., and Jain, A. Deep Learning Driven Natural Languages Text to SQL Query Conversion: A Survey. *arXiv preprint arXiv:2208.04415*, 2022. Available at: `https://arxiv.org/abs/2208.04415`

[11] Mohammadjafari, A., Golshan, B., and Tan, W.-C. From Natural Language to SQL: Review of LLM-based Text-to-SQL Systems. *arXiv preprint arXiv:2410.01066*, 2024. Available at: `https://arxiv.org/abs/2410.01066`

[12] Wong, A., Ho, S. S., Ooi, B. C., Tan, K.-L., Yao, A. C., Yiu, M. L., Zhang, M., Zheng, B., Zheng, K., and Zhu, F. Translating Natural Language Queries to SQL Using the T5 Model. *arXiv preprint arXiv:2312.12414*, 2023. Available at: `https://arxiv.org/abs/2312.12414`

[13] Yang, A. Z. H., Martins, R., Menezes, C., Steinhardt, J., Abbeel, P., and Stoica, I. On Repairing Natural Language to SQL Queries. *arXiv preprint arXiv:2310.03866*, 2023. Available at: `https://arxiv.org/abs/2310.03866`