Large Language Models for Generative Information Extraction a Survey

Transforming Unstructured Text into Structured Information with LLMs

M MOHSIN SIDDIQI

Outline

Authors: Derong XU, Wei CHEN, Wenjun PENG, Chao ZHANG, Tong XU, Xiangyu ZHAO, Xian WU, Yefeng ZHENG, Yang WANG, Enhong CHEN

Journal: Frontiers of Computer Science (2024), Volume 18, Issue 6, Article 186357

Link: https://link.springer.com/content/pdf/10.1007/s11704-024-40555-y.pdf

Introduction to IE
Understanding the data challenge and defining IE

Universal IE Frameworks
Consolidating NER, RE, and EE under one model

Evolution of Methodologies
From traditional pipelines to generative paradigms

NL vs Code LLMs
Comparing natural language and code-focused models

Generative IE Paradigm
Formulation and tasks: NER, RE, and EE

Optimization Techniques
Prompt engineering, data augmentation, and more

Looking Back and Forward
BERT's role, challenges, and future directions

The Data Challenge

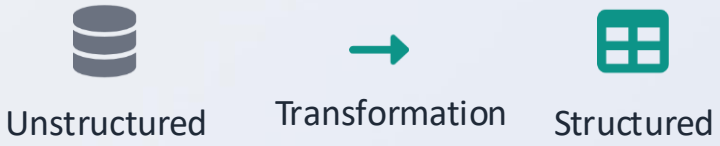
The digital age has led to an unprecedented explosion of explosion of unstructured data, while organizations struggle to struggle to transform this raw information into valuable, valuable, structured formats.

The Transformation Challenge

- Most unstructured data lacks inherent structure
- Manual extraction is time-consuming and error-prone error-prone
- Organizations need automated solutions for computational analysis

Types of Unstructured Data

- Text Documents
- Web Pages
- Social Media
- Emails



Enter Large Language Models

LLMs offer powerful new capabilities to automate the extraction of meaningful information from unstructured data, unlocking latent value within massive datasets.

What is Information Extraction (IE)?

Information Extraction (IE) is the task of **automatically identifying** and **extracting structured information** from unstructured or semi-structured machine-readable documents.

Unstructured Text

Free-form text paragraphs, articles, reports...

IE Process

Structured Output

Database, knowledge graph, or predefined entities/relations/events

Conversion Process

Transforms free-form text into structured, queryable formats

Computational Analysis

Enables automated analysis of information that would require manual effort

Foundation for AI

Cornerstone technology for deriving actionable insights from textual data

Key Goals of Information Extraction

Identifying Entities

Recognizing and classifying named entities in text into predefined categories:

- Person names (John Smith)
- Organizations (Apple, Inc.)
- Locations (New York, USA)
- Time expressions (Yesterday, 2023)
- Monetary values ($50, €25)

Extracting Relations

Discovering semantic relationships between identified entities:

- "John Smith" **works at** "Apple, Inc."
- "Apple, Inc." **is located in** "Cupertino"
- "Yesterday" **is related to** "2023"
- "$50" **represents** "price"

Detecting Events

Identifying occurrences of specific events and their associated participants:

- Corporate earnings announcements
- Product launches
- Personnel changes
- Financial transactions

Important and Applications of IE

Information Extraction transforms unstructured text into structured data, enabling computational analysis across diverse domains.

Financial Analysis

Extracting company names, stock prices, financial indicators, and market trends from news articles and reports.

Medical Record Processing

Identifying diseases, symptoms, treatments, medications, and patient demographics from clinical notes.

Customer Feedback Analysis

Extracting product features, sentiment, and common complaints from reviews, surveys, and social media.

Legal Document Review

Identifying parties, clauses, obligations, and dates from contracts and legal filings.

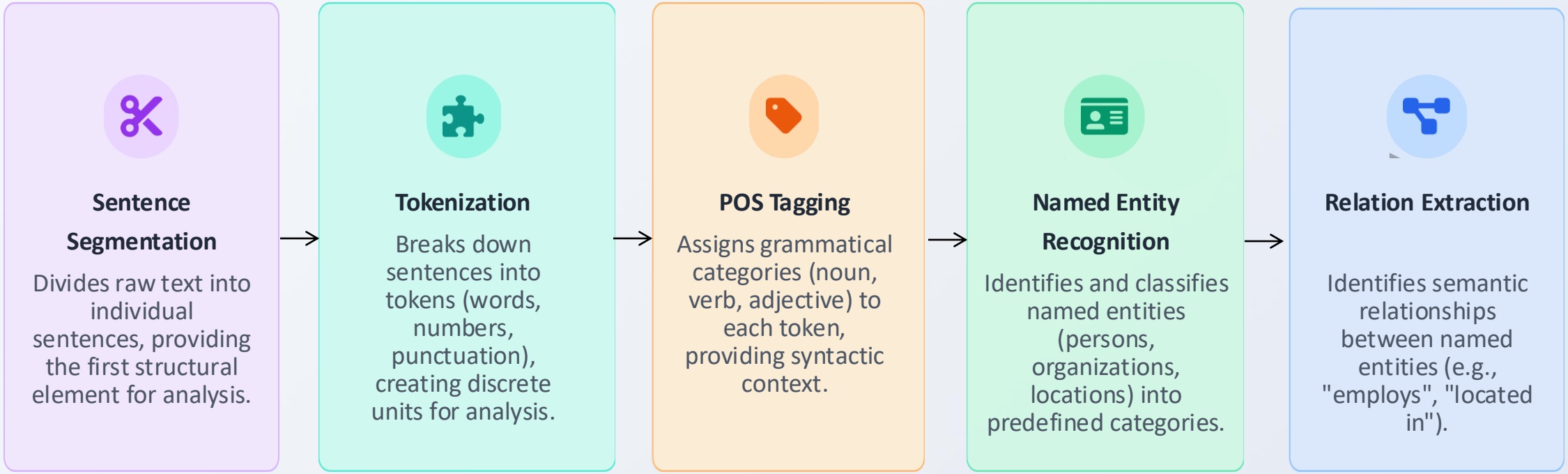
Competitive Intelligence

Extracting competitor activities, product launches, and strategic partnerships from public sources.

Intelligence & Security

Identifying threats, actors, locations, and timelines from intelligence reports and open-source information.

IE Pipeline Steps



Limitations of Traditional Methods

Error Propagation

Errors in early stages (tokenization, NER) cascade through subsequent stages, compounding inaccuracies and reducing overall system performance.

Poor Generalization

Models trained on one domain or dataset often perform poorly when applied to new domains due to reliance on domain-specific features and patterns.

Heavy Reliance on Feature Engineering

Requires extensive manual feature engineering by domain experts, which is time-consuming, labor-intensive, and demands deep linguistic knowledge.

Lack of End-to-End Learning

The modular nature prevents end-to-end optimization, as each component is typically optimized independently, potentially missing global dependencies.

High Costs for Labeled Data

Training robust models often demands large amounts of meticulously labeled data, which is expensive and difficult to obtain, especially for new domains.

Traditional pipelines suffer from error propagation through sequential stages

The Generative Paradigm Shift

Traditional IE Pipeline

- Multi-stage, sequential processing
- Task-specific models for each component
- Discriminative steps for identification
- Requires extensive feature engineering
- Cascading errors between stages

Reframing as Generation

Sequence-to-sequence or text-to-text generation

Unified Framework

End-to-end approach with a single model

Generative IE Approach

- Direct generation of structured output
- Emergent reasoning capabilities
- Flexibility across diverse tasks
- Reduced need for feature engineering
- Zero-shot and few-shot learning

Generative IE Formulation

Generative IE reframes traditional IE tasks as **text generation problems** , using a single LLM to directly generate structured output from unstructured text.

Input Text

"Barack Obama was born in Honolulu..."

LLM Generator

Single model handles all IE tasks through text generation

Structured Output

JSON, XML, or other structured formats

Unified Framework

Consolidates NER, RE, and EE under one model

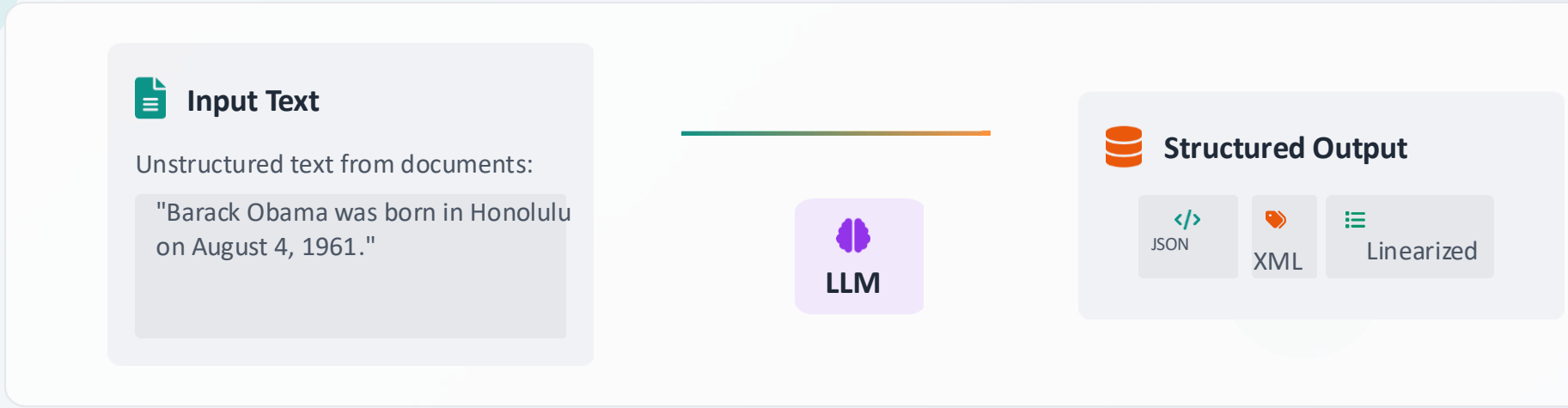
Text-to-Structure

Converts text to JSON, XML, or linearized formats

Emergent Properties

Leverages LLM's emergent reasoning capabilities

Text-to-Structure Transformation



JSON Format

```
{
"entities": [
{"name": "Barack Obama", "type": "person"},
{"name": "Honolulu", "type": "location"},
"location"},
{"name": "August 4, 1961", "type": "date"}
"date"}
]
}
```

XML Format

```
<document>
<entity type="person">Barack Obama</entity>
<entity type="location">Honolulu</entity>
>
<entity type="date">August 4, 1961</entity>
</document>
```

Linearized Format

(Barack Obama/PERSON) (was/BELONGS) (Honolulu/LOCATION) (Barack Obama/PERSON) (was/BELONGS) (August 4, 1961/DATE)

Generative IE reframes IE as a **text-to-structure** task

Named Entity Recognition (NER) in Generative IE

Key Concept

In generative IE, LLMs perform NER by directly generating entity mentions with their classifications classifications from input text, eliminating the need need for specialized models.

How It Works

- LLMs read text and generate structured entity entity outputs
- Entities are classified using natural language prompts
- No separate NER model required - universal universal frameworks apply

"Barack Obama was born in Honolulu and served as the 44th President of the United States."

Input Text → **LLM Processing** → **Entity Generation**

Example Output:

[Barack Obama] - PERSON
[Honolulu] - LOCATION
[44th President] - ROLE

Benefits in Generative IE

- Eliminates need for task-specific models
- Reduces error propagation from pipelines

Relation Extraction (RE) with LLMs

Generative RE Approach

- Reframes RE as **text-to-text generation** problem
- Identifies semantic relationships between entities in unified format
- Leverages model's knowledge for context understanding

Key Advantage

End-to-end extraction without need for task-specific models

RE Process with LLMs

Barack Obama was born in Honolulu.

Extract the relationship between the entities:

Relationship:
born in

Entities:
Barack Obama → Honolulu

Text-to-Text Format

Input:
Extract relationships from: "Barack Obama was born in Honolulu"

→

Output:
[{"relation": "born in", "subject": "Barack Obama", "object": "Honolulu"}]

Event Extraction (EE) Capabilities

What is Event Extraction?

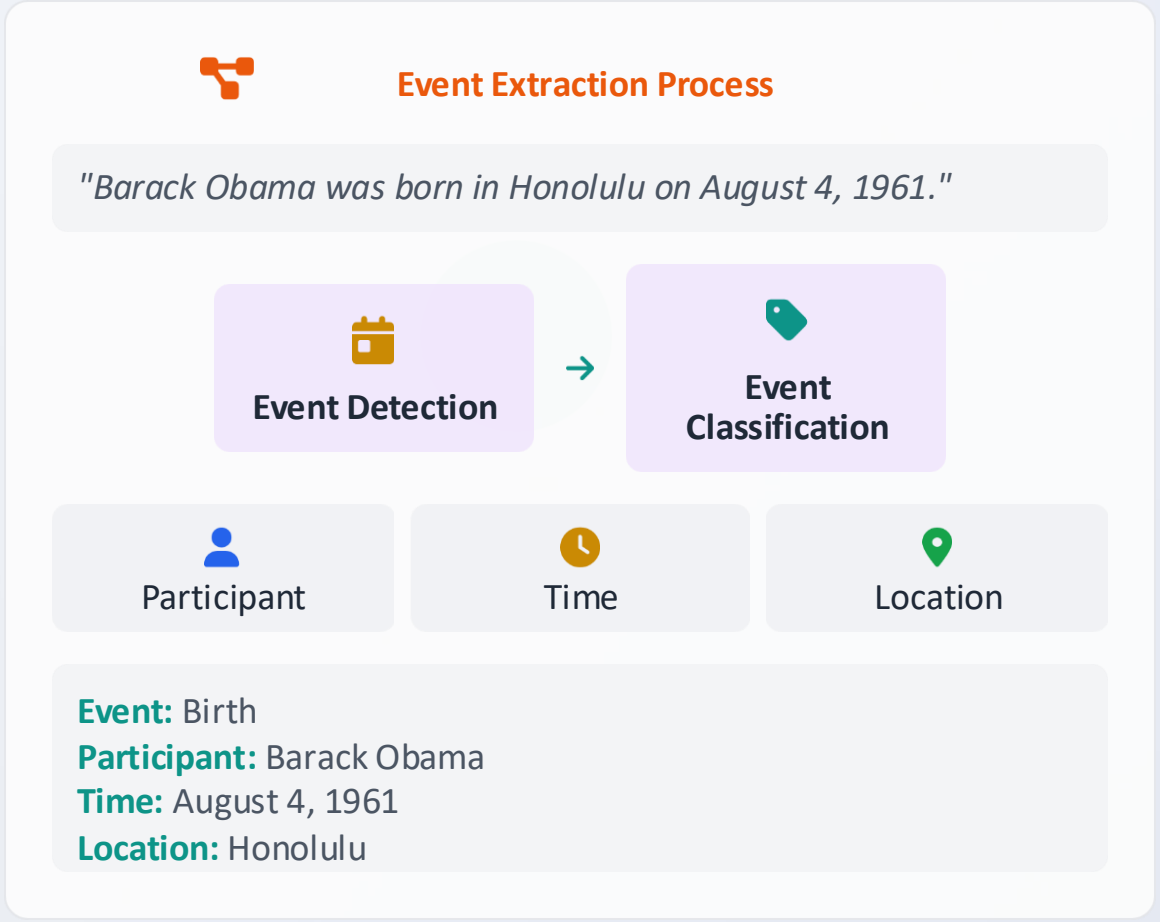
Event Extraction identifies occurrences of specific events and their associated participants, time, and location through generative text processing.

Generative Approach

LLMs detect events by generating structured outputs that identify event types, participants, and temporal/spatial information from unstructured text.

Key Advantages

- Unified framework for extracting events and arguments
- Zero-shot and few-shot learning capabilities
- Adaptability to new event types without retraining



Universal IE Frameworks

Unified LLM Framework

Generative model handling all IE tasks

Named Entity Recognition

Extracting entities and classifications

Relation Extraction

Finding relationships between entities

Event Extraction

Identifying events and participants

Unified Approach

- Consolidates NER, RE, and EE under a single model
- Converts diverse IE tasks into text-to-text generation
- Uses consistent interface for various extraction needs

Key Advantages

- Adapts to new extraction tasks with minimal fine-tuning
- Reduces need for task-specific models
- Enables simultaneous extraction of entities, relations, and events

Natural Language vs Code LLMs

Comparison for Information Extraction tasks

Natural Language LLMs

- **Strengths:** Better at understanding context and nuance in text
- **Strengths:** More intuitive for extractive tasks
- **Weaknesses:** Less consistent output structure
- **Best for:** Tasks requiring deep text comprehension

Example Applications

- Named Entity Recognition
- Relation Extraction
- Event Extraction

Code LLMs

- **Strengths:** More structured and consistent output
- **Strengths:** Better at following specific formatting instructions
- **Weaknesses:** May struggle with complex text understanding.
- **Best for:** Tasks requiring precise output structure

Example Applications

- JSON-formatted data extraction
- Code generation for data processing
- Structured output formatting

Data Augmentation Techniques

Methods to enhance training data quality and quantity for improved generative IE performance

Textual Rewriting

Generate paraphrases of existing training examples to increase diversity and coverage

Multi-Source Integration

Combine data from multiple sources to increase diversity and improve model robustness

Back-Translation

Translate text to another language and back to create variations while preserving meaning

Confidence Filtering

Select and amplify high-confidence predictions from model-generated data

Self-Augmentation

Use a teacher model to generate additional training examples from unlabeled data

Temporal Sampling

Incorporate data from different time periods to improve model's generalization

Prompt Engineering Strategies

Techniques for optimizing LLM performance in Information Extraction tasks

Zero-shot Prompting

- No task-specific examples provided
- Model relies on built-in knowledge
- Quick setup with generic instructions

Best for: Quick prototyping or when examples are scarce

Few-shot Prompting

- Provides 1-5 examples
- Shows input-output pairs
- Adapts model to specific IE task

Best for: Specific IE tasks with clear examples

Chain-of-Thought Prompting

- Requests step-by-step reasoning
- Breaks complex tasks into steps
- Improves accuracy for complex IE

Best for: Complex IE tasks requiring reasoning

Comparison of Prompting Strategies

Analyzing different prompting approaches for generative IE tasks.

Strategy	Description	Complexity	Performance
Zero-shot	Direct prompting without examples	●●○	●●○
Few-shot	Providing 1-5 examples	●●●	●●●
Chain-of-Thought	Step-by-step reasoning	●●●	●●●

Resource Requirements

- Zero-shot: Lowest
- Few-shot: Medium
- Chain-of-Thought: Highest

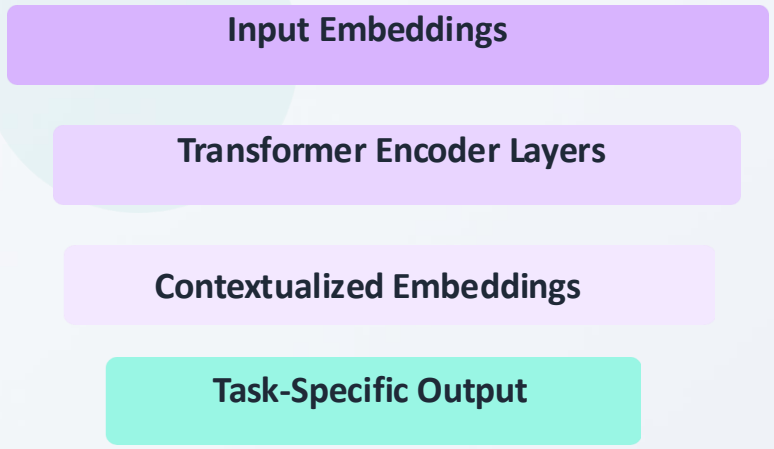
Task Suitability

- Zero-shot: Simple IE
- Few-shot: Medium complexity
- Chain-of-Thought: Complex reasoning

Implementation

- Zero-shot: Easiest
- Few-shot: Medium difficulty
- Chain-of-Thought: Most complex

BERT's Role in Information Extraction



BERT (Bidirectional Encoder)

Foundational Role

Encoder-only architectures like BERT played a foundational role in advancing NLP and IE before generative models.

Contextualized Embeddings

BERT revolutionized IE by providing rich contextualized embeddings that understand word nuances and relationships.

Feature Extraction

Served as a powerful feature extractor for IE tasks like NER and RE, significantly improving performance.

Current Relevance

Remains highly effective for specific IE tasks, serving as a strong baseline against which generative approaches are compared.

When BERT Still Matters

- Computational efficiency
- Fine-grained discriminative power
- Specific IE task optimization

BERT vs Generative Models

BERT-based Approaches

- Encoder-only architectures providing contextualized embeddings
- Powerful feature extraction for words and phrases
- Strong performance on specific IE tasks like NER and RE
- Efficient computational requirements
- Often serve as strong baselines for comparison

Generative LLM Approaches

- End-to-end solutions reframing IE as text-to-text generation
- Zero-shot and few-shot learning capabilities
- Reduced need for extensive labeled datasets
- Emergent reasoning capabilities for complex tasks
- Unified framework handling various IE tasks

Key Differences & Implications

Task Focus

BERT: Discriminative, finding specific elements
LLMs: Generative, creating structured output

Architectural Approach

BERT: Multiple specialized models needed
LLMs: Single flexible model for various tasks

Current Challenges in Generative IE

Key obstacles in LLM-based information extraction

Hallucinations

Models sometimes produce factually incorrect information, critical for accuracy-sensitive applications.

Factual Consistency

Ensuring extracted information aligns with source text and external knowledge bases.

Output Structure Control

Precisely controlling output format (e.g., JSON schema) remains challenging for complex tasks.

Computational Costs

Training and deploying large models are resource-intensive, affecting accessibility and real-time applications.

Data Scarcity for Low-Resource Languages/Domains

Performance degrades in domains with limited training data, despite few-shot capabilities.

Future Directions and Research

Exploring the next frontiers in generative Information Extraction

Multi-modal IE

Research focuses on extracting information from a combination of text, images, and other data types, expanding IE capabilities beyond text analysis.

Efficient Architectures

Developing more efficient models for real-time extraction, reducing computational costs while maintaining accuracy for practical applications.

Improved Explainability

Enhancing the ability to understand and explain LLM-generated extractions, making the "black box" more transparent for critical applications.

Trustworthiness

Developing methods to improve factual consistency and reduce hallucinations in LLM-generated extractions for high-stakes applications.

Benefits of Generative IE

Unified Frameworks

Consolidates diverse IE tasks (NER, RE, EE) under a single model, eliminating the need for multiple task-specific models and simplifying the extraction pipeline.

Reduced Feature Engineering

Leverages pre-trained knowledge and emergent reasoning capabilities, significantly reducing the manual feature engineering required for traditional IE approaches.

Few-shot Capabilities

Enables zero-shot and few-shot learning across diverse domains and tasks, drastically reducing the need for extensive labeled datasets.

Improved Adaptability

Offers unprecedented flexibility and generalization across diverse domains and IE objectives, making it highly adaptable to new tasks with minimal fine-tuning.

Key Advantage: Generative IE simplifies complex IE pipelines into unified, end-to-end generation tasks

Conclusion and Key Takeaways

"Generative Information Extraction represents a significant paradigm shift, simplifying traditional IE pipelines into unified, end-to-end generation tasks."

Paradigm Shift

Moving from traditional multi-stage pipelines to a unified generative approach.

Flexibility

Zero-shot and few-shot learning capabilities reduce need for labeled datasets.

Accessibility

Making IE more accessible across diverse applications and domains.

- **Text-to-Structure Transformation:** Converting unstructured text into structured formats like JSON, XML, or tables.
- **Universal Frameworks:** Consolidating NER, RE, and EE under a single, flexible model architecture.
- **Prompt Engineering:** Critical technique for guiding LLMs to perform specific IE tasks effectively.
- **Future Directions:** Multi-modal IE, efficient architectures, and improving explainability of LLM-generated extractions.

THANK YOU