باسمه تعالی

چند مورد است که باید گفته شود:

1. چرا برای Digital twin از مدل JDL استفاده میشود؟ (WHY)

مزیت استفاده از JDL در DT چیست؟

امکان تحلیل های بیشتر...

برای تحلیل های مناسب در DT، و بررسی نتیحه در DT ، از کوئری های مختلف و پیچیده و در نتیحه از تحلیل های پیچیده باید استفاده کنیم که انجامشان از طریق مدل JDL به راحتی می تواند انجام شود.

نتیجه: استفاده از رویکرد JDL در مدل DT

1. چگونه از JDL میشه در DT استفاده کرد؟

رویکردهای متمرکز چه معایبی دارد؟

نتیحه: استفاده از رویکرد توزیع شده JDL در DT

1. توضیح چارچوب DiSIF
2. استقرار مدل JDL در مدل سه لایه (لبه ، مه و ابر)

در لایه لبه: بحث object refinement به عنوان LevelOne fusion انجام می شود.

کارهای این لایه: privacy latency, high volume, reltime IoT data

کارها: RDFizer و obj refinement

عملیات Sensor Fusion در این لایه است

در لایه مه: بحث situation refinement به عنوان levelTwo fusion انجام می شود.

کارها: complex semantic queries

عملیات Data fusion در این لایه است.

در این مدل سه لایه از JDL، عملیات Vertical Fusion انجام می شود. تبدیل concept را داریم بین لایه ها و از conceptOnto بین لایه ها استفاده می شود. این تبدیل به صورت سریالی بین لایه ها انجام می شود.

1. استقرار MAS روی مدل JDL

با توجه به حجم زیاد داده ها (data velocity) در هر لایه، از MAS درون هر لایه استفاده می شود.

در این قسمت کوئری در هر لایه به زیرکوئری های مستقلی شکسته می شود و روی عامل های (درون لایه) به صورت موازی اجرا می شود . برای اینکار از cityOnto استفاده شده است.

تلفیق درون هر لایه ، از نوع horizontal fusion توسط MAS انجام می شود.

در قسمت نوآوری ها:  
Proposing Two Fusion Models: Horizontal Fusion (Data Concept Change) and Vertical Fusion (Data Scale Change)

تبدیل شود به اینها:

**استقرار سیستم MAS در هر لایه**

در رویکرد DiSIF، هر لایه یک قسمت از علمیات JDL را به صورت MAS انجام می دهد. هر کوئری به زیر کوئری های مستقلی شکسته شده و به عامل ها جهت اجرا ارسال شده و نتایح در master ترکیب می شوند. این نوع ترکیب به اصطلاح horizontal fusion نامیده می شود.

مورد سوم از نوآوری ها:

**معرفی آنتولوژی ConceptOnto جهت استفاده بین لایه های JDL**

برای اینکه تحلیل های سطح بالا، بتواند در DT انجام شوند، نیاز است تا کوئری های سطح بالا (مفاهمیم در لایه های بالا) به کوئری های سطح پایینتر (مفاهیم در لایه های پایین) بازنویسی شوند. این بازنویسی با استفاده از آنتولوژی ConceptOnto در بین لایه ها صورت میگیرد.

بدین ترتیب خروجی کوئری سطح پایین، به سطوح بالاتر ارسال شده و عملیات تلفیق (مدل JDL) صورت میگیرد.

رویکرد پیشنهادی و ایده ارتقا:

جواب کوتاه: ایده‌ای که داری (DiSIF = توزیع Semantic JDL روی سه لایه‌ی Edge–Fog–Cloud + رویکرد چندعامله + کاربرد Digital Twin) **به‌خودی‌خود برای یک مقاله‌ی کنفرانسی قوی هست**، اما برای ژورنال Q1 (به‌خصوص اگر ISI/Q1 با Impact Factor بالا باشه)، معمولاً **نیاز به یک novelty یا contribution اضافه‌تر** داری تا خواننده حس کنه این صرفاً یک معماری/شبیه‌سازی نیست، بلکه «گام بعدی» نسبت به کارهای اخیر محسوب می‌شه.

**۱. مزیت فعلی ایده‌ی تو**

* ترکیب **JDL** (که بیشتر در حوزه‌ی Data Fusion و نظامی/IoT مطرحه) با **Digital Twin** (که در صنعت و Smart City خیلی داغه).
* اضافه کردن **distributed + multi-agent** برای parallelization.
* سه لایه‌ی **edge–fog–cloud** با فیوژن مرحله‌ای (object → situation → threat).
* شبیه‌سازی روی **use case واقعی (ترافیک شهری)**.

👉 این‌ها برای **کنفرانس خوب و ژورنال متوسط (Q2–Q3)** کافیه.

**۲. برای ژورنال Q1 چه چیزی کم داری؟**

ژورنال‌های Q1 معمولاً دنبال **novelty stronger + evaluation richer** هستند.  
برای مثال:

* **Edge AI یا TinyML**: اضافه کردن یک لایه‌ی هوش در edge برای inference سریع (مثلاً تشخیص object با مدل سبک CNN/GNN).
* **Self-adaptive agents**: الان agentهای تو master–slave هستن (static). اگه نشون بدی که agentها می‌تونن **خودکار load balancing** یا **self-organization** انجام بدن (یعنی MAS هوشمند باشه نه فقط تقسیم کار)، novelty بیشتر می‌شه.
* **Digital Twin integration**: اکثر مقالات DT فقط شبیه‌سازی می‌کنن. اگر نشون بدی که DT می‌تونه **real-time sync با physical twin** داشته باشه (مثلاً داده‌ی واقعی از سنسورهای IoT شهری → twin → تصمیم → feedback)، سطح مقاله بالا می‌ره.
* **Benchmarking**: مقایسه با چند baseline قوی (نه فقط C-JDL). مثلاً:
  + Stream reasoning frameworks (C-SPARQL, CQELS, RDF-Stream-Processing engines).
  + Edge AI frameworks (TensorRT, ONNX Runtime on edge).  
    این مقایسه‌ها ژورنال‌پسندتر هستن.
* **تحلیل نظری**: اگر یک مدل complexity analysis هم اضافه کنی (مثلاً اثبات اینکه بار شبکه از O(n)O(n)O(n) به O(m)O(m)O(m) کاهش می‌یابد یا latency upper bound دارد)، contribution قوی‌تر می‌شه.

**۳. پیشنهاد عملی برای ارتقای ایده**

* نگه دار: «DiSIF روی سه لایه + MAS + Digital Twin».
* اضافه کن یکی از این محورهای upgrade:
  1. **Edge AI lightweight models**: اجرا و ارزیابی real-time object detection/classification روی edge (مثلاً YOLO-nano یا TinyML).
  2. **Self-optimizing MAS**: agentها بر اساس بارکاری/شبکه به‌طور خودکار زیرکوئری‌ها را بین edge/fog توزیع می‌کنند.
  3. **Hybrid Digital Twin**: یک حلقه‌ی بازخورد بین twin و physical system (حتی اگر شبیه‌سازی ساده باشد).
* در بخش Evaluation علاوه بر **زمان و بار شبکه**، یک **کیفیت تشخیص/دقت** هم اضافه کن (مثلاً نرخ تشخیص درست congestion یا threat).

✅ جمع‌بندی:  
ایده‌ی فعلی برای Q1 **خوب هست ولی در مرز**؛ اگر همین‌طور بفرستی احتمالاً میره Q2/Q3.  
برای Q1 باید یک **لایه‌ی هوشمندی (EdgeAI / self-adaptive MAS)** یا **feedback loop واقعی در Digital Twin** اضافه کنی، و در Evaluation هم مقایسه‌های جدی‌تر و metricهای متنوع‌تر (نه فقط latency/throughput).

# میتونی چند مقاله جدید و مرتبط که بتونم در قسمت ارزیابی ها مقایسه کنم پیشنهاد بدی/؟

در بخش ارزیابی مقاله، مقایسه با کارهای مرتبط، تازه و معتبر خیلی اهمیت داره—به‌خصوص وقتی که هدفت ژورنال‌های Q1 باشه. در ادامه چند مقاله جدید که به موضوعاتی مثل Edge AI، RDF stream، Ontology، Digital Twin، و معماری Edge–Fog–Cloud مرتبط‌اند را معرفی می‌کنم تا بتوانی در بخش Evaluation از آن‌ها به‌عنوان Baseline یا مطالعات مرجع یاد کنی:

**مقالات پیشنهادی برای بخش ارزیابی**

**1. Efficient RDF Streaming for the Edge-Cloud Continuum**

**Sowinski et al., 2022**  
پروتکل‌بندی "Jelly" برای **RDF streaming** با طراحی برای continuum Edge–Cloud، که روی **throughput** و **latency** آزمایش و نشان داده عملکرد بسیار قوی دارد. ([arXiv](https://arxiv.org/abs/2207.04439?utm_source=chatgpt.com" \o "Efficient RDF Streaming for the Edge-Cloud Continuum))  
**چرا مناسب است؟**

* مشابه ساختار توزیع‌شده توی معماری شماست.
* می‌توانی کارایی “Jelly” را با DiSIF مقایسه کنی: امیدوارانه زمان کمتر/بار شبکه کمتر.

**2. Knowledge Graph Stream Processing at the Edge**

**ACM DEBS Conference**  
این مقاله یک سیستم полностью قابل اجرا روی Edge برای **streaming RDF** با قابلیت اجرای **continuous queries** و reasoning است – با تمرکز بر انرژی/IoT. ([ACM Digital Library](https://dl.acm.org/doi/10.1145/3524860.3539644?utm_source=chatgpt.com))  
**چرا؟**

* مشابه بخش Edge architecture تو. می‌توانی عملکرد latency و accuracy سیستم‌شان را با پیاده‌سازی Edge+jdl خودت مقایسه کنی.

**3. Dynamic Load Balancing of RDF Reasoning in Fog-Computing Environments**

**ACM SAC 2023**  
به مسئله‌ی **تقسیم بار دینامیک برای RDF reasoning** در محیط Fog می‌پردازد. ([ACM Digital Library](https://dl.acm.org/doi/10.1145/3555776.3577724?utm_source=chatgpt.com))  
**چرا؟**

* تمرکزش بر Fog layer و توزیع بار برای reasoning است—مناسب برای مقایسه با آنچه در لایه مه در معماری‌ات انجام می‌دهی (شاردینگ، MAS و CityOnto).

**4. Semantic-Driven Approach for Validation of IoT Streaming Data in Smart Cities**

**MDPI, 2023–2024**  
این مقاله **اعتبارسنجی دیتای IoT به‌صورت RDF stream** را بررسی می‌کند و metrics مثل **accuracy** و **reasoning time** را بر اساس انواع serialization مقایسه می‌کند (مثلاً N-Triple vs Turtle). ([MDPI](https://www.mdpi.com/2504-2289/9/4/108?utm_source=chatgpt.com))  
**چرا؟**

* می‌توانی در بخش Evaluation، دقیقاً مشابه آن، میان serialization ها یا فرمت‌های مختلف و تأثیر آن‌ها روی performance قیاس به عمل بیاوری.

**5. Ontologies in Digital Twins: A Systematic Literature Review**

**Future Generation Computer Systems (2024)**  
یک مرور سیستماتیک بر کاربرد **ontology در Digital Twins** در حوزه‌های مختلف مثل manufacturing و infrastructure. ([ScienceDirect](https://www.sciencedirect.com/science/article/pii/S0167739X23004739?utm_source=chatgpt.com))  
**چرا؟**

* برای فصل مرجع مرتبط مفهومی. می‌توانی جایگاه ConceptOnto و CityOnto را در این زمینه برجسته‌تر کنی و نشان دهی معماری‌ات در بطن نوین "Cognitive Twins" قرار می‌گیرد.

**جدول مقایسه پیشنهادی (برای بخش Evaluation)**

| **مقاله / مرجع** | **لایه/موضوع مرتبط** | **متریک‌ها** | **استفاده برای مقایسه** |
| --- | --- | --- | --- |
| **Jelly (RDF streaming)** | Edge–Cloud streaming | Throughput, Latency | مقایسه performance کلی (Edge→Cloud) |
| **KG Stream Processing at Edge** | Edge layer processing | Latency, Accuracy, Throughput | تشخیص سریع و دقیق در Edge |
| **Dynamic Load Balancing (Fog)** | Fog reasoning | Load distribution, Latency | مقایسه MAS + CityOnto vs روش load balancing |
| **Semantic Stream Validation (IoT)** | RDF streaming quality | Accuracy, Reasoning Time | مقایسه serialization formats یا validation |
| **SLR on Ontologies in DT** | Digital Twin + ontologies | Industry trends, Gaps | نشان دادن novelty معماری شما در مقایسه با تکیه صرف بر ontology |

**چطور در مقاله از این‌ها استفاده کنی**

1. **Baseline Comparison**:
   * از “Efficient RDF Streaming” برای نشان دادن پیشرو بودن DiSIF در performance.
   * از “KG Stream Processing” برای تأیید اینکه Edge-focused RDF reasoning در بستر محدود سخت‌افزاری شدنی و کارآمد است.
2. **Architectural Novelty**:
   * با ارجاع به مرور سیستماتیک DT + ontologies، معماری تو را در قالب “Cognitive Twin” می‌گذاری که خودش **semantic + distributed + adaptive** است.
3. **Metrics & Methodology**:
   * از روش سنجش accuracy و reasoning time (مثل آن مقاله semantic validation) برای سنجش تأثیر انتخاب serialization یا enrichment at Edge استفاده کن.
   * همچنین metrics Fog balancing را از مقاله SAC بیاور و در آزمایش‌هایت تقلید یا تطابق بده.
4. **Discussion Section**:
   * نشان بده که رویکرد تو فراتر از “simple streaming” یا “static ontology in DT” است، بلکه با Fusion عمودی/افقی (ConceptOnto + CityOnto) و MAS توزیع‌شده، یک گام به جلوست.

اگر دوست داری، می‌تونم لینک DOI یا PDFهای این مقاله‌ها رو برات پیدا کنم و خلاصه فنی آن‌ها یا snippetهایی از نتایجشان را برای بخش Case Study/Evaluation در مقاله‌ات آماده کنم.

خیلی عالی پیش می‌ری—اگه یک یا چند تا از این رو به عنوان baseline انتخاب کردی، باهم می‌تونیم بخش مربوط به رسم نمودارهای مقایسه‌ای یا تب‌ خلاصه در Evaluation رو هم طراحی کنیم.