KeXtract

JSON SCHEMA FOR KEXTRACT

Versione	1.1
Stato:	RELEASED
Data:	20/01/2026

Kedos Srl - Information Technology Solutions
DIVISIONI OPERATIVE
Viale Monza, 1 - 20125 Milano (MI)
C.so Inghilterra, 49 - 10138 Torino (TO)
Strada G. Inzani, 5 - 43125 Parma (PR)

SEDE LEGALE: via Chiavari, 5/E - 43125 Parma (PR),
P.IVA 02629980349, - R.I. PR-254346 Capitale sociale: 220.000,00€ i.v. - **Tel. 0236741057 - Fax 0521.995819**

Guida alla scrittura di JSON Schema per output strutturato in KeXtract

A. Introduzione

Questa guida fornisce linee guida pratiche per la scrittura di JSON Schema efficaci destinati all'estrazione strutturata di dati da documenti tramite KeXtract.

L'obiettivo è aiutare sviluppatori e data engineer a progettare schemi che bilancino chiarezza e concisione, massimizzando l'accuratezza dell'estrazione senza sovraccaricare il contesto del modello.

B. JSON Schema

JSON Schema rappresenta il formato ottimale per guidare l'estrazione strutturata di dati da documenti tramite KeXtract per diverse ragioni fondamentali:

- **Linguaggio dichiarativo standard**: JSON Schema è uno standard industriale (IETF draft) ampiamente adottato, con semantica ben definita e tool di validazione consolidati. Questo garantisce interoperabilità, manutenibilità e una curva di apprendimento ridotta per sviluppatori già familiari con l'ecosistema JSON.
- **Struttura come contratto**: Lo schema funge da contratto esplicito tra l'utente e il sistema di estrazione. Definisce con precisione la forma dell'output atteso, i tipi di dati, i vincoli di validazione e le relazioni gerarchiche. Questo elimina ambiguità e permette validazione deterministica dell'output.
- **Bilanciamento tra espressività e semplicità**: JSON Schema offre costrutti sufficientemente ricchi da modellare strutture dati complesse (oggetti annidati, array, composizioni, enumerazioni) mantenendo una sintassi leggibile e compatta. Questo è cruciale per non saturare il budget di attenzione di KeXtract.
- **Ottimizzazione del contesto**: A differenza di formalismi più verbosi (es. descrizioni in linguaggio naturale), JSON Schema codifica vincoli strutturali in modo estremamente compatto. Ogni token risparmiato nella definizione dello schema libera risorse cognitive del modello per processare il contenuto del documento stesso.
- **Validazione programmatica**: L'output prodotto da KeXtract può essere validato automaticamente contro lo schema tramite librerie standard, garantendo robustezza e facilitando l'integrazione in pipeline di produzione. Errori strutturali vengono rilevati immediatamente senza necessità di parsing custom.
- **Separazione di responsabilità**: Lo schema si concentra sulla *struttura* dei dati, mentre KeXtraxct si concentra sulla *comprensione semantica* del documento. Questa separazione di responsabilità è fondamentale per ottenere estrazioni accurate e deterministiche. Quando si definisce uno schema JSON per guidare KeXtract nell'estrazione dell'output, l'obiettivo è bilanciare chiarezza e concisione. Lo schema deve essere sufficientemente descrittivo da guidare il modello verso output consistenti, ma non così verboso da confondere o appesantire il contesto.

Best Practices

0. Approccio iterativo e verifica multi-documento

La costruzione di uno schema JSON efficace per l'estrazione dati richiede un processo empirico e incrementale. Non è possibile prevedere a priori tutte le variazioni di layout, terminologia e struttura che i documenti reali presenteranno. Per questo motivo, è fondamentale adottare un ciclo di sviluppo basato su test continui e raffinamenti progressivi:

- Lavorare un campo alla volta: iniziare con i campi essenziali e aggiungerli progressivamente
- Testare ogni nuovo campo con almeno 3-5 documenti di tipologie diverse prima di procedere
- Verificare la consistenza dell'estrazione su tutti i formati di input possibili nel proprio progetto (PDF, immagini, scansioni)
- Raffinare le `description` in base ai risultati osservati sui diversi documenti
- Evitare di costruire schemi completi "a tavolino" senza validazione empirica

1. Essere descrittivi, non prolissi

La chiarezza nella denominazione e descrizione è essenziale per guidare KeXtraxct verso estrazioni accurate. Ogni elemento dello schema dovrebbe comunicare il suo scopo in modo immediato e inequivocabile, senza sovraccaricare il contesto con informazioni ridondanti. Si raccomanda di:

- Usare nomi di proprietà auto-esplicativi (`userEmail` invece di `e`)
- Le `description` devono aggiungere valore, non ripetere il nome della proprietà
- Evitare frasi ridondanti: preferire "Data di creazione dell'entità" a "Questa proprietà contiene il valore della data in cui l'entità è stata creata"

2. Descrivere il contesto e il dominio del documento

Quando si lavora con documenti specialistici o di settori specifici, è fondamentale fornire una descrizione esplicita del contesto e del dominio nel campo `description` dello schema principale o delle proprietà chiave. Questo previene che KeXtract deduca alcune informazioni sulla propria conoscenza di addestramento, che potrebbe però essere generica o non allineata con la terminologia specifica del vostro dominio.

Per documenti tecnici o di settori verticali, specificare:

- Il tipo di documento e il suo scopo (es. "Referto di analisi cliniche di laboratorio", "Contratto di locazione commerciale")
- Le convenzioni terminologiche specifiche del settore (es. "Nel contesto assicurativo, 'premio' indica l'importo pagato dal contraente")
- Eventuali standard o normative di riferimento (es. "Fattura elettronica conforme al formato XML FatturaPA")

Questa contestualizzazione esplicita è interamente a vostro carico e rappresenta il valore aggiunto che solo voi, conoscendo il dominio applicativo, potete fornire per guidare l'estrazione verso risultati accurati e consistenti.

3. Comprendere i limiti dei modelli

Ogni modello LLM ha quantità limitata di "spazio cognitivo" che può dedicare ai token del contesto durante l'elaborazione. A differenza della memoria umana, che può rileggere e focalizzarsi su parti specifiche di un testo, il modello deve processare tutti i token contemporaneamente, questo meccanismo ha un costo computazionale crescente: ogni token deve "guardare" tutti gli altri token del contesto, e il modello ha un numero finito di risorse da distribuire. Per cui, quando il prompt diventa troppo lungo o complesso, i token entrano in competizione per ottenere attenzione, e il modello inizia a perdere capacità di seguire vincoli strutturali precisi.

Gli effetti pratici di un budget saturo includono:

- Parti importanti dello schema vengono ignorate
- Aumenta la probabilità di errori strutturali nell'output
- La fedeltà allo schema diminuisce progressivamente
- Vincoli complessi (nidificazioni profonde, enum articolati) vengono disattesi

Per questo motivo, schemi JSON concisi e ben strutturati non sono solo una questione di eleganza, ma una necessità tecnica: ogni token di `description` ridondante sottrae risorse che KeXtract potrebbe dedicare alla comprensione del documento stesso.

4. Sfruttare le description

Le `description` sono lo strumento principale per comunicare con KeXtract e guidarlo verso estrazioni accurate. A differenza dei nomi delle proprietà, che devono essere concisi e machine-readable, le description possono essere più discorsive e ricche di contesto. Ogni description ben scritta riduce l'ambiguità semantica e aumenta la probabilità che il modello identifichi correttamente il dato target nel documento.

- Aggiungere `description` a ogni proprietà per fornire contesto al modello
- Specificare vincoli o aspettative (es. "Valore compreso tra 0 e 100")
- Indicare esempi quando utile: "Formato: +39 123 4567890"

Disambiguare valori simili

Quando un documento contiene più valori numerici o testuali che potrebbero rispondere allo stesso nome generico, è fondamentale essere espliciti nella `description` per guidare KeXtract verso il dato corretto. Ad esempio, una fattura può contenere diversi importi (subtotali parziali, imponibile, IVA, totale finale), e una proprietà chiamata semplicemente `totalAmount` risulta ambigua. In questi casi, la `description` deve specificare con precisione quale valore estrarre, facendo riferimento alla terminologia esatta utilizzata nel documento o alla sua posizione logica. Preferire descrizioni come "Importo totale finale della fattura, IVA inclusa" o "Subtotale imponibile, esclusa IVA" piuttosto che generiche come "totale" o "importo". Questo principio si applica a qualsiasi categoria di dati che possa presentarsi in forme multiple: date (emissione vs scadenza vs pagamento), identificativi (numero documento vs numero ordine vs riferimento cliente), quantità (pezzi vs colli vs unità di misura).

5. Tipizzare accuratamente i dati

La tipizzazione corretta dei dati è essenziale per garantire che l'output strutturato di KeXtract sia immediatamente utilizzabile dal proprio sistema senza necessità di parsing o conversioni aggiuntive. Una tipizzazione accurata riduce gli errori di interpretazione, facilita la validazione e permette al modello di concentrarsi sull'identificazione semantica del dato piuttosto che sulla sua rappresentazione formale.

- Usare i tipi primitivi appropriati: `string`, `number`, `integer`, `boolean`, `array`, `object`
- Specificare `enum` quando i valori possibili sono limitati
- Usare `type: ["string", "null"]` per valori opzionali invece di omettere il tipo

6. Specificare sempre il formato

Specificare il formato dei dati è fondamentale per due ragioni: garantisce un output uniforme e prevedibile che può essere processato deterministicamente dal sistema downstream, e consente a KeXtract di concentrare le sue risorse computazionali sulla comprensione semantica del contenuto piuttosto che sull'inferenza del formato corretto. Quando il modello sa esattamente quale formato è atteso (es. una data in formato ISO 8601), può focalizzarsi sull'identificazione del valore corretto nel documento, riducendo ambiguità e migliorando l'accuratezza dell'estrazione.

- Per date: `"format": "date"` (YYYY-MM-DD)
- Per timestamp: `"format": "date-time"` (ISO 8601)
- Per orari: `"format": "time"` (HH:MM:SS)
- Per email: `"format": "email"`
- Per URI: `"format": "uri"`

7. Contestualizzare oggetti e array

Quando si lavora con strutture dati complesse come oggetti e array, è essenziale fornire contesto sufficiente affinché KeXtract comprenda non solo la struttura formale dello schema, ma anche il significato semantico di ogni elemento e la sua relazione con il resto del documento. Una contestualizzazione chiara migliora significativamente l'accuratezza dell'estrazione, specialmente quando si tratta di dati ripetitivi o gerarchici.

- Per array, usare `items` per definire lo schema degli elementi
- Specificare `minItems` e `maxItems` quando rilevante
- Per oggetti nested, fornire `description` che spieghi la relazione con l'oggetto parent
- Usare `additionalProperties: false` per prevenire campi non previsti

8. Lavorare per contesti, nidificando gli oggetti

La nidificazione di oggetti nello schema JSON non è solo una questione di organizzazione formale, ma riflette la struttura logica e semantica del documento stesso. Raggruppare proprietà correlate in oggetti nested permette a KeXtract di comprendere meglio il

contesto di estrazione e le relazioni tra i dati, migliorando significativamente l'accuratezza. Tuttavia, è fondamentale bilanciare la profondità della nidificazione con le capacità di attenzione del modello: strutture troppo complesse possono degradare le performance.

Suggerimenti:

- Organizzare lo schema in contesti logici che rispecchiano la struttura del documento
- Usare oggetti nested per raggruppare campi correlati (es. `"customer": { "name", "email", "address" }`)
- Limitare la nidificazione a 2-3 livelli: strutture troppo profonde riducono l'affidabilità dell'estrazione
- Preferire array di oggetti semplici piuttosto che singoli oggetti con molti campi opzionali
- Ogni livello di nidificazione deve avere una `description` che chiarisca il contesto e la relazione con il parent

Esempio di buona nidificazione:

```
{
  "invoice": {
    "type": "object",
    "description": "Dati principali della fattura",
    "properties": {
      "number": { "type": "string", "description": "Numero fattura" },
      "customer": {
        "type": "object",
        "description": "Anagrafica cliente destinatario",
        "properties": {
          "name": { "type": "string" },
          "vatNumber": { "type": "string" }
        }
      },
      "items": {
        "type": "array",
        "description": "Righe di dettaglio della fattura",
        "items": {
          "type": "object",
          "properties": {
            "description": { "type": "string" },
            "amount": { "type": "number" }
          }
        }
      }
    }
  }
}
```

9. Usare le capacità visuali di KeXtract

KeXtract supporta modelli multimodali in grado di elaborare sia il contenuto testuale che la struttura visiva del documento. Questo permette di sfruttare riferimenti spaziali e layout per migliorare l'accuratezza dell'estrazione, specialmente su documenti con formattazione complessa o ambigua.

Suggerimenti per sfruttare le capacità visuali:

- Nelle `description`, includere riferimenti alla posizione visiva quando rilevante (es. "Totale finale, tipicamente nell'angolo in basso a destra del documento")
- Specificare caratteristiche di formattazione distintive (es. "Importo evidenziato in grassetto o con font maggiorato")
- Indicare relazioni spaziali tra campi (es. "Data di emissione, generalmente posizionata vicino al numero fattura nell'intestazione")
- Per tabelle, descrivere la struttura attesa (es. "Colonna 'Quantità' tipicamente a sinistra della colonna 'Prezzo unitario'")
- Menzionare elementi visivi distintivi come box, linee di separazione, o sezioni colorate che delimitano aree specifiche

Questo approccio è particolarmente utile per:

- Distinguere tra valori numerici simili in posizioni diverse (es. subtotali parziali vs totale finale)
- Estrarre dati da tabelle complesse con molte colonne
- Identificare campi in documenti con layout non standardizzato

- Gestire documenti scansionati dove la qualità OCR potrebbe essere variabile

10. Evitare calcoli e relazioni tra elementi

KeXtract è uno strumento di estrazione, non di elaborazione. La sua funzione principale è identificare e trascrivere dati presenti nel documento, non eseguire operazioni su di essi. Delegare calcoli o validazioni al modello introduce variabilità, riduce l'affidabilità e rende difficile il debugging. Un approccio corretto prevede l'estrazione di dati atomici e l'esecuzione di tutte le elaborazioni deterministiche nel codice applicativo.

- Non chiedere a KeXtract di eseguire conteggi, somme o altre operazioni aritmetiche sui dati estratti
- Non richiedere validazioni incrociate o relazioni tra campi (es. "verificare che la somma delle righe corrisponda al totale")
- Estrarre solo i valori atomici presenti nel documento: calcoli e validazioni devono essere eseguiti programmaticamente nel sistema downstream
- L'approccio deterministico post-estrazione è più affidabile e manutenibile rispetto a far eseguire operazioni a KeXtract

Esempio da evitare:

```
{
  "totalAmount": {
    "type": "number",
    "description": "Somma degli importi di tutte le righe"
  }
}
```

Approccio corretto:

```
{
  "items": {
    "type": "array",
    "items": {
      "type": "object",
      "properties": {
        "amount": { "type": "number", "description": "Importo della singola riga" }
      }
    }
  },
  "declaredTotal": {
    "type": "number",
    "description": "Totale dichiarato nel documento"
  }
}
```

Estrarre sia le righe individuali che il totale dichiarato permette di calcolare e validare la somma programmaticamente.

11. Suddividere l'estrazione per documenti complessi

Per documenti con struttura articolata o contenuti eterogenei, un singolo schema omnicomprensivo può saturare il budget di attenzione del modello e ridurre l'accuratezza complessiva. In questi casi, è preferibile adottare una strategia di estrazione modulare, suddividendo il documento in contesti logici indipendenti ed effettuando chiamate separate con schemi focalizzati. Questo approccio migliora significativamente la precisione, semplifica il debugging e rende lo schema più manutenibile nel tempo.

- Per documenti particolarmente articolati o con molte sezioni eterogenee, considerare l'approccio di estrazioni multiple
- Effettuare chiamate separate per ogni contesto logico del documento (es. anagrafica, righe dettaglio, totali, note)
- Ogni chiamata utilizza uno schema focalizzato su un sottoinsieme specifico di informazioni
- Questo approccio riduce la complessità dello schema, migliora l'affidabilità dell'estrazione e semplifica il debugging
- Ricomporre i dati estratti programmaticamente nell'applicazione downstream

Esempio: per una fattura complessa, invece di un unico schema monolitico, effettuare 3 chiamate separate:

1. Schema per dati anagrafici (emittente, destinatario, riferimenti documento)
2. Schema per righe di dettaglio

3. Schema per totali e imposte

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "extractedDate": {
      "type": "string",
      "format": "date-time",
      "description": "Timestamp ISO 8601 dell'estrazione"
    },
    "entities": {
      "type": "array",
      "description": "Lista delle entità identificate nel testo",
      "items": {
        "type": "object",
        "properties": {
          "name": {
            "type": "string",
            "description": "Nome dell'entità"
          },
          "type": {
            "type": "string",
            "enum": ["person", "organization", "location"],
            "description": "Categoria dell'entità"
          },
          "confidence": {
            "type": "number",
            "minimum": 0,
            "maximum": 1,
            "description": "Livello di confidenza (0-1)"
          }
        },
        "required": ["name", "type"]
      }
    }
  },
  "required": ["extractedDate", "entities"]
}
```