* Layers (RDF4J Sails):
* Semiotic Layer: Verbs, Relationships.
* DOM Layer: Dynamic Object Model. CSPO Materialized Semiotic Layer.
* Sets Layer: DOM Layer CSPO Resource Arrangement.
* Augmentations / FCA Layer: Sets Layer FCA / Embeddings.
* Functional Layer: API.
* Semiotic Layer:
* Verbs: action (rel end: amante) / passion (rel end: amado) / state (rel: amor: ama / amado). Relation parts attributes.
* Semiotics: CSPO Schema. DOM Statements Source.
* Occurrences: Objects.
* Concepts: Non Terminals.
* Signs: Terminals.
* Semantics / Pragmatics:
* (Context, Occurrence, Sign, Concept);
* (Context, Occurrence, Concept, Sign);
* Grammar:
* (Context, Concept, Occurrence, Sign);
* (Context, Concept, Sign, Occurrence);
* Syntax:
* (Context, Sign, Concept, Occurrence);
* (Context, Sign, Occurrence, Concept);
* DOM Layer:
* (Class, Instance, Attribute, Value);
* Sets Layer:
* Sets Encoding / Arrangements:
* Subject, Predicate, Object Sets inside Context Set (CSPO Statements Resources Populated)
* SubjectKind Set: Statements Predicates / Objects intersection.
* PredicateKind Set: Statements Subjects / Objects intersection.
* ObjectKind Set: Statements Predicates / Subjects intersection.
* Kinds Population (Subject Example): Aggregate Subjects Occurrences with same Predicates (type), same Objects (instance).
* Kind Reification (Subject Example): S: (SubjectKind, Subject, Predicate, Object);
* Resource Monad: For each input SPO URNs, prime sequence identifying each new Resource. Resource registry.
* Resource Embeddings: set of Resources Occurrences (Statements) Contexts / Attributes products.
* Statements: SPO / FCA (Object, Context, Attribute) Resource triples.
* Kind Monad: For each Kind set of matching Attributes / Values, set of matching Resources. Identifier (reification) Resource: Embedding URN.
* CSPO quads Statements Context: Kind (inheritance).
* Resource<URN>
* Resource<Statement>
* Resource<Kind>
* SubjectKind: Kind;
* PredicateKind: Kind;
* ObjectKind: Kind;
* Resource<URN>::map(ctx::getSubjectOccurrences) : Set<Resource<Statement>>;
* Resource<URN>::map(ctx::getPredicateOccurrences) : Set<Resource<Statement>>;
* Resource<URN>::map(ctx::getObjectOccurrences) : Set<Resource<Statement>>;
* Resource<Statement>::map(ctx::getSubjectKind) : Resource<Kind>;
* Resource<Statement>::map(ctx::getPredicateKind) : Resource<Kind>;
* Resource<Statement>::map(ctx::getObjectKind) : Resource<Kind>;
* Resource<Kind>::map(ctx::getKindContexts) : List<Resource<URN>>;
* Resource<Kind>::map(ctx::getKindAttributes) : List<Resource<URN>>;
* Resource<Kind>::map(ctx::getKindValues) : List<Resource<URN>>;
* Augmentations / FCA Layer:
* FCA:
* FCA Attributes: Primes / one-hot Bitstring Encoding.
* FCA Context: Concepts / Objects Attributes.
* Embeddings: CSPOs FCA Contexts Objects URNs Attributes Primes Product / Bitstring OR.
* Vector Space Model: CSPO Dimensions. CSPO Points: Objects URN Embeddings.
* Similarity / Distance: Common FCA Embedding Attributes Factors. Common Super Concept / Object. VSM Vector Similarity.
* Transforms / Translation: Object, Object. Merge Attributes, extract Similarity on merged Subject and merged Objects (Transforms).
* State (flow): Attributes index (hasAddress), values (address: xyz) masks browsing (Concepts / Objects).
* Ontology Matching: Concept Lattice shape.
* Augmentations:
* Generate Embeddings from DOM SPO URNs FCA Contexts (Context(Attributes, Objects)).
* FCA Contexts: Map Attributes URNs with primes sequence / one-hot bitstring and Objects URNs with theirs Attributes product / bitstring OR Embedding. Embed Context (SPO / Kinds) into Contexts Attributes (product).
* SPO URNs Resources Wrapper (Contexts Attributes / Objects Embeddings Population / Encoding). Subject URN Embedding Encoding Example: Aggregate all Subject Embedding Contexts Occurrences Concept Attributes Primes Product.
* Subject Embeddings: Contexts: Predicates; Attributes: Objects; Objects: Subjects;
* Predicate Embeddings: Contexts: Objects; Attributes: Subjects; Objects: Predicates;
* Object Embeddings: Contexts: Subjects; Attributes: Predicates; Objects: Objects;
* SubjectKind Embeddings: Contexts: Subjects; Attributes: Predicate Contexts (occurrences for Subject); Objects: Object Contexts (occurrences for Subject);
* PredicateKind Embeddings: Contexts: Predicates; Attributes: Subject Contexts (occurrences for Predicate); Objects: Object Contexts (occurrences for Predicate);
* ObjectKind Embeddings: Contexts: Objects; Attributes: Predicate Contexts (occurrences for Object); Objects: Subject Contexts (occurrences for Object);
* Schema Aggregation: Type (Kind) in Context / Role Inference. Align Attributes with existing Kinds. CSPO Embeddings Clustering. Kinds Naming / Labels: Alignment Embedding Encoded Placeholders.
* Data Alignment: Align / Reify Instances with Kinds. Complete (align) Type (Kind) Instance Attributes Values in Context / Role Inference. Embeddings Features zero-shot Classification (Aggregation Kinds Embedding Encoded Labels). Missing Values (links) Prediction: Placeholder Embeddings with resolvable context metadata.
* Behavior Activation: Performed / Available Resource State (DIDs) Transforms. Perform Prompt: Aggregate / Align. Response: Embeddings Context Facts / Next Available Prompts (Alignment). Embeddings Features Regression (Prompts Suggestions).
* Embeddings Features Average (User Embedding, Product Embedding) Predictions.
* Functional API:
* Runtime: RDF4J.
* REST Browsing State Based Dialog Wizard. DOM / DCI / CDI / Augmentations (FCA). Structured Prompts / Responses (Statements Flow, Relationships / Roles).
* URNs: Semantic Identifiers. Encoding. DIDs (Distributed IDs). Angular Encoding.
* Conversational State Transfer (COST): Distributed (P2P).
* Node.js: JSON-LD. Functional Resources (Monads, run-at: request client / server peer. Context State available Functors / Transforms).
* Parsing: Transforms.