From Paper to Production: Tracking the AI Research-to-Implementation Gap

Final Report

Knowledge and Data 2025 - Group 80

Nia Racheva - 2841982
Anastasia Ciubucova - 2856242
Kareem Tujjar - 2863031
Dylan He - 2800209

Date: 27-10-2025

1 Introduction

1.1 Goal, motivation and users

Primary Goal:
This proposal quantifies the disparity between research and implementation of AI/ML in the context of a timeline from academic publishing to implementation. We examine how long it will take before AI innovation (models, architectures, approaches) between research articles and production repositories, as well as factors that Transmigrate that facilitate or impede diffusion.

Scope:
We concentrate on models published during 2020-2024 for deep learning, studying their development via:
- Scholarly articles, or 'Papers with Code'.
- Activity on GitHub repo (stars, forks, releases)
- Framework adoption patterns (PyTorch, TensorFlow, or JAX, etc
- Benchmarks for Breakthrough Innovation Performance

Motivation:
Motivation for this project comes from the increasing speed of AI innovation, where breakthrough research can be production-ready in months—or languish in obscurity within academia. What this gap indicates is that it enables informed investment, technology, or research direction decisions for stakeholders.

With the inclusion of data from GitHub, DBpedia, Wikidata, and Papers With Code, a comprehensive knowledge graph helps us extract tangible patterns for spreading AI innovation from academia to industries.

Principal audience and users:
- AI/ML engineers: Determining suitable directions for research that are production-ready
- Technology companies - Timing of investment in new AI solutions
- Researchers and teachers - making an investment decision for innovation implementation

1.2 Competency Questions

1. What time lag exists between AI-related academic studies and implementation in reality?
2. Which AI frameworks are adapted the fastest in industry in comparison to academia?
3. What features set groundbreaking AI models apart from other smaller updates?

This question speaks to research to practice gaps, that is, our project. CQ1, our main question, measures time relations that defines publications, repositories, and releases in order to illustrate adoption over time. CQ2 studies framework-specific adoption patterns based on repository statistics, Publication metrics. CQ3 provides innovation metrics via benchmark results and architecture features. Altogether, such questions enable our targeted group of users to inform technology adoption, investment, and curriculum development decisions with data insights that illuminate tangible AI/ML know-how transfer trends.

2 External sources identified

2.1 Existing Ontologies Selected

- **Schema.org.**

Schema.org is a structured markup language developed by Google, Microsoft, Yandex, and Yahoo! to help website owners create richer and more accurate search engine results. The Schema.org page describes a myriad of “Hobbies and Leisure” types to choose from. Source: https://schema.org/. The selected ontology is Schema.org for modeling of software artifacts and research outputs. We reused the following classes and properties. We mapped the schema:SoftwareApplication to our Framework class to represent ML platforms. Our Release class becomes the mapper for versioned software releases with schema:SoftwareRelease. schema:CreativeWork that accompanies our Publication class for research papers. Schema name date is our published publication date . By using these alignments, we were able to align existing web data seamlessly and temporal analysis of the research-timeline-implementation trajectory. We imported Schema.org vocabulary by declaring the prefix in our ontology file. We were able to type our classes as subclass or equivalent where required e.g. Repository rdfs:subClassOf schema:SoftwareApplication.

- **Friend of a Friend (FOAF).**

FOAF is a simple vocabulary that allows to describe people and their activities in social or collaborative contexts. As well as their relationships with other people, and with things. Source: http://xmlns.com/foaf/spec/. FOAF has been chosen as a model to represent researchers as well as organizational entities in the AI/ML ecosystem We reused. This foaf:Person is mapped to our Person class for researchers and engineers. An organization that falls under our Organization class: research labs and companies. Properties include foaf:name, foaf:workplaceHomepage, foaf:made.Thanks to these vocabularies, we were able to relate a researcher to his publications and an organization to the frameworks that it maintains. This helped us construct provenance data, which sheds light on the role of institutions in the adoption of AI. We imported the FOAF vocabulary (expressed in Turtle) into our ontology with an owl prefix declaration and used the owl:equivalentClass statement to say that our Person and Organization classes were equivalent to the respective FOAF classes. We manually checked all alignments to validate them by looking at the class definitions in imported ontologies and ensuring their consistency with our competency questions. We verified that the inferences made by the reasoner (e.g. associating authors with their organisations) worked as expected in Protege.

2.2 External SPARQL endpoints identified

DBpedia
DBpedia is a crowd-sourced community effort to extract structured information from Wikipedia and make this information available on the Web. It’s chock full of billions of triples and all the other datasets you might want to query for AI-related things.
Source: https://dbpedia.org/sparql.
Motivation: DBpdaedia contains a large amount of information that is great for finding some information about AI/ML frameworks, researchers or companies. We can query it to discover facts, like who created a tool or when did it launch — and observe how AI tech evolves with the passage of time.

Wikidata
Wikidata is a large collaboratively built knowledge graph in RDF containing more than 100 million items. You've got lots of metadata related to software, researchers and papers in the form of SPARQL queries to drill down into relationships and timelines.
Source: https://query.wikidata.org/
Motivation: Wikidata for accurate and high-quality data we can use to map the AI/ML Tools landscape, creators of these tools, as well track over time (e.g., version updates). It is great for connecting researchers with their work and identifying trends in AI advancement.

2.3 External non-RDF Datasets identified

We chose two datasets (JSON and CSV) that are not RDF to make the knowledge graph more comprehensive. These will be converted to RDF to interlink with our other sources and visualize AI/ML trends.

Papers with Code

Papers with Code is a dataset that associates machine learning publications with code implementations. It is available via a JSON API. It catalogs papers, datasets and frameworks, along with metadata like authors, date of publication, and GitHub link.

Source: https://paperswithcode.com/api.

Motivation: This dataset is a treasure trove for our project, as it links AI/ML publications to the tools they are implemented with. We could use this as a way to see the models - and frameworks - that are popular and being promoted - perfect for building our knowledge graph.

GitHub Archive

GitHub Archive is a project to record the public GitHub timeline, archive it, and make it freely available for later analysis. It captures development trends of the AI/ML software hosted on GitHub, with the amount of stars and forks from the timestamp of a popular repository.

Source: https://www.gharchive.org/.

Motivation: Tons of AI/ML tools are built on GitHub, so this data allows us to see activity, popularity, and contributors over time. With it in RDF format, we can connect tools to researchers and understand how AI software frameworks improve.

3 Design of Ontology

3.1 Methodology

We worked in loops and focused on questions competence. First, we defined.
We will analyze the difference between AI ML research and practice, including stakeholders. They need to understand the adoption timelines. “How quickly can academic AI research reach usability?” was the key competency question we developed. And two other auxiliary questions surrounding the Framework adoption patterns and innovation indicators. Next, we used a reuse-first strategy.
Whenever feasible, we utilize established vocabularies. An example of the vocabulary we use: for software and creative works, we use Schema.org, and for people and organizations, we use the FOAF vocabulary. This way, our graph is able to integrate external data properly.
We then developed a concept model that represents the following and their relationships (e.g., a Model utilizes a Framework, implements an Architecture, and has BenchmarkResults measured on Datasets):

- Framework
- Model
- Architecture
- Task
- Dataset
- Benchmark
- BenchmarkResult
- Publication
- Repository
- Release
- Organization
- Person

For population and alignment, we plan to retrieve entities from DBpedia/Wikidata and metrics from PapersWithCode and GitHub with external identifiers linked to our IRIs via sameAs relations. Finally, we reason over the filled graph using a reasoner (class consistency) and simple SHACL validation (e.g., obligatory properties and numeric ranges), and loop on failure or insufficient coverage of the competency questions.

3.2 Conceptualisation

3.2.1 Classes

Class	**Brief description**	**Key alignments / notes**	**Disjoint with**
Framework	ML software platform	schema:SoftwareApplication	Model, Dataset, Publication
Model	A concrete trained model	might reuse wd:Q3966 instances when available	Dataset, Person
Architecture	Abstract design pattern (CNN, Transformer, Diffusion)	maps from Wikidata “architecture” items	Framework, Dataset
Task	Problem type (Image)	could align to PWC tasks	–
Dataset	Training/evaluation dataset	dcat:Dataset compatible	Framework, Publication
Benchmark	Named benchmark/leaderboard	–	–
BenchmarkResult	Result of evaluating a model on a dataset/benchmark	analytical node	–
Publication	Paper introducing a model/technique	dbo:AcademicArticle	Organization
Repository	Code repository	prov:Entity	Publication

Release	Time-stamped software/model release/version	schema:SoftwareRelease	Publication
Organization	Company, lab, university	org:Organization/foaf:Organization	Person
Person	Researcher/engineer	foaf:Person	Organization

3.2.2 Properties

Object Properties

Property	**Domain → Range**	**Inverse**	**Notes / characteristics**
implementsArchitecture	Model → Architecture	implementedBy	Functional (exactly one; see Restriction R1)
usesFramework	Model → Framework	usedByModel	Many-to-many
addressesTask	Model → Task	–	Helps answer “which techniques solve which problems”
evaluatedOn	BenchmarkResult → Dataset	hasEvaluation	Functional (see R2)
forBenchmark	BenchmarkResult → Benchmark	–	–
ofModel	BenchmarkResult → Model	–	–
introduces	Publication → (Model OR Framework)	introducedBy	–
hasRepository	(Model OR Framework) → Repository	isRepoOf	–
hasRelease	Repository → Release	–	Enables time series
maintainedBy	Framework → Organization	maintains	–
authoredBy	Publication → (Person OR Organization)	authorOf	reuse dcterms:creator as alias

Data Properties

Data property	**Domain**	**Range**	**Example**
stars	Repository	xsd:integer	GitHub stars
forks	Repository	xsd:integer	GitHub forks
citations	Publication	xsd:integer	Crossref/GS counts

releaseDate	Release	xsd:date	2023-07-18
score	BenchmarkResult	xsd:decimal	89.7
metricName	BenchmarkResult	xsd:string	“accuracy”, “F1”, “MMLU”

3.2.3 Restrictions

ID	Description	OWL expression (informal)	Rationale
R1	Each Model implements exactly one Architecture	Model subclassof (=1 implementsArchitecture.Architecture)	Keeps modeling consistent: “ResNet-50 implements CNN”, “GPT-3 implements Transformer”
R2	Each BenchmarkResult must refer to exactly one Dataset and at least one metric	BenchmarkResult subclassof (=1 evaluatedOn.Dataset) And (SOME achievesMetric.PerformanceMetric)	Ensures results are interpretable and comparable
(optional) R3	Release must connect to exactly one Repository	Release subclassof (=1 isReleaseOf.Repository)	For clean time-series per repo

3.3 Formalization / Implementation

The ontology is implemented in OWL 2 and should be submitted as a Turtle file. We use a common base namespace for our terms and re-use FOAF and Schema.org to gain maximum interoperability. The class hierarchy follows the pipeline of analysis: Framework and Model occupy a central position, with design families being abstracted by Architecture; Problem settings are characterised by Task and Dataset; Evaluations are modelled by Benchmark andBenchmarkResult; Publication, Repository and Release relate research outputs and software artefacts; Organisation and Person provide provenance. The key object properties that link these classes: a Model instantiates an Architecture, operates based on a Framework, and is for a Task; a BenchmarkResult is run on a Dataset and for benchmark links; Publications present Models or Frameworks; Repositories contain Models/Frameworks with releases. Data properties store quantitative signals like repository stars or forks, benchmark scores and release dates. We also add two lightweight cardinality constraints: 1 - each Model entails precisely one Architecture, consistent with the way papers position a model in a family of designs; 2 - each BenchmarkResult is related to exactly one Dataset in order to avoid uncertain scores. The other properties remain optional in order to admit incomplete web data.
It takes care of the alignment and identity by instantiating IRIs from source ids and stating owl:sameAs/schema:sameAs to relevant Wikidata/DBpedia/PapersWithCode/GitHub entities.

4 Data Integration and conversion

Data Integration and Conversion

The integration process unified distinct data into a single Knowledge Graph, incorporating our proposed ontology. By reusing and aligning established vocabularies — Schema.org, FOAF, and DBpedia Ontology — we ensured interoperability and consistency between research and implementation data.

Ontology Alignment

Each ontology was applied to a specific semantic scope.
Software entities were modeled using schema:SoftwareApplication and schema:SoftwareRelease.
People and organizations were represented through FOAF classes.
Scholarly works, such as research papers, were aligned with dbo:AcademicArticle from the DBpedia Ontology.
Mappings were connected via owl:sameAs, allowing identical resources to be related across DBpedia, Wikidata, and our instances.

Data Conversion

Non-RDF datasets — Papers with Code (JSON) and GitHub Archive (CSV) — were converted into RDF using Python and rdflib.
The conversion of Papers with Code data generated Publication and Model instances connected through introduces and addressesTask.
GitHub data populated Repository and Release entities with quantitative data properties such as stars, forks, and releaseDate.

External RDF Sources

Mashups were created using federated SPARQL queries to retrieve AI-related information — frameworks, creators, and release dates — from DBpedia and Wikidata. Retrieved URIs were normalized and linked to local IRIs to maintain referential consistency.

Reasoning and Validation

After merging all triples, the HermiT reasoner was applied to check logical consistency and infer new relationships, such as linking models to their publishers through publications.
SHACL constraints were used to validate key rules: each Model implements exactly one Architecture, and each BenchmarkResult refers to exactly one Dataset.
The integration of academic and production data in the Knowledge Graph enables meaningful insights into how AI research transitions into practical implementation.

5 The resulting knowledge graph

After the data integration and conversion process ontology final_1.ttl the whole knowledge graph successfully developed. The graph combines research papers, AI models,

frameworks, architectures, datasets, benchmarks, and repositories into a single structure that traces the route from idea to implementation.
You can see the complete knowledge graph in the Jupyter notebook (implementation.ipynb).
Overall size: *16,664 RDF triples*

5.1 Overview

The knowledge graph created reflects the conceptual relations and practical development links in the AI ecosystem. This indicates that every research paper showcases one or more AI models and demonstrates how they are produced using architect and frameworks to their eventual availability in public repositories with official releases. The graph, which visually resembles a network, consists of nodes, which refer to an object (a model or a dataset), and edges, which refer to a relationship. 'PyTorch', 'TensorFlow', 'Keras' : Refers to the software environment in which the model is built and trained on.
Frameworks → Organizations.
Frameworks are programs maintained or developed by organizations such as research groups or tech companies (Meta AI, Google Brain, etc.).

5.2 Structure and Relationships.

The AI research cycle elements are linked in the knowledge graph through a clear semantic model.
Publications → Models.
Every published paper has at least one ML model. This represents the academic origin of the innovation.
Models → Architectures.
Every model is based on a certain architecture like transformer architecture or Convolutional Neural Network. This captures the technical foundation of the model.
Models → Frameworks.
By frameworks (PyTorch, TensorFlow, Keras), it is meant the software environment in which the model has been coded and trained.
Frameworks → Organizations.
Frameworks fall under software either maintained by research organizations, corporations (for instance, Meta's AI, Google's Brain), or their initiatives. Models → Tasks. All models have their own specific function with respect to one or more tasks like image classification, machine translation, or text generation.
Models → Repositories → Releases.
code repositories like GitHub once models are implemented. Every Repository Has Many Releases To Mark Their Development Over The Time.
Models → Datasets.
The performance of models is evaluated on datasets. These datasets are often connected and have benchmark results that report scores and metrics.
The relationships described result in complete chain linking research idea to implemented systems and demonstrates how academic innovation becomes operational technology.

5.3 Statistics and Composition.

The integrated graph contains 16,664 unique RDF triples, comprising more than 160 unique entities and 18 types of relationships. It includes:

- 4 models.

- 4 frameworks.
- 3 architectures.
- 3 publications.
- 3 repositories with 6 releases.
- 2 benchmark results.
- Around 20 persons and organizations.

All entities follow the naming convention from the base namespace http://example.org/ai-research-gap# . The graph also offers external alignment links to DBpedia and Wikidata through the use of owl:sameAs.

5.4 Interpretation.

The knowledge graph helps in identifying the actors and their relationships in the AI research to implementation ecosystem.
We can analyze the speed at which frameworks are adopted, the influence of certain architectures on model performance and research ideas that migrate into production software.
This visualisation combines structured academic data with real world development indicators (e.g. GitHub stars, forks and release dates) enabling the knowledge diffusion to be visualised in AI.
We will use this integrated model as our basis for reasoning, inference, and SPARQL querying in the upcoming sections.

6 Meaningful Inferences

6.1 Inferred Relationships

1. Associating publications with institutions via authorship.

By combining the FOAF vocabulary with the DBpedia vocabulary, the reasoner concluded affiliations for researchers who authored particular research papers. By way of illustration, the authors who wrote “Attention Is All You Need” were associated with Google Brain, indicating the organization behind the research work and the TensorFlow library alike.

2. Linking Models to Organizational Maintenance.

Through the property chain

Model → usesFramework → maintainedBy → Organization

3. Inferring Benchmark Participation.

When a BenchmarkResult is associated with a model and a dataset, an inference could be made from the ontology that the respective model is participating in a specific benchmark, for instance, since ResNet-50 has results associated with ImageNet, it is inferred to participate in the ImageNet Classification Benchmark.

4. Identifying architecture families.

By linking each model to exactly one architecture (R1 restriction), the reasoner grouped models into families. All ResNet instances were classified as Convolutional Neural Networks (CNN), while Transformer and GPT-3 fell under the Transformer architecture family. This allows for comparing the speed of adoption across different architectural types.

5. Detecting real-world implementation evidence.

Combining GitHub release data with publication metadata provided insights into which academic models were implemented in production. Models with linked repositories and release dates were automatically recognized as implemented, offering a measurable indicator of research moving to production.

6.2 Interpretation

These insights confirm the semantic consistency of the graph and turn it from a static dataset into an analytical tool. They reveal hidden relationships, including institutional influence, framework spread, and model reuse, which help explain how innovation transitions from paper to product. In Protégé, the reasoner displayed these as new object-property assertions under the inferred view, confirming the success of the ontology design and the logical soundness of the combined knowledge base.

7 Relevant SPARQL queries

The knowledge graph integrates data from Papers with Code, GitHub, DBpedia, and Wikidata, leveraging our schema.org ontology (final_1.ttl) to track the AI research-to-implementation gap. Below are three SPARQL queries we designed to address the competency questions. We utilised notable features such as FILTER, GROUP BY, and optional patterns for richer insights.

CQ1: Time Lag Between Academic Studies and Implementation

```
PREFIX ex: <http://example.org/ai-research-gap#>
PREFIX schema: <https://schema.org/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?publication ?pubDate ?release ?relDate
      (YEAR(?relDate) - YEAR(?pubDate) AS ?yearDiff)
WHERE {
  ?publication a ex:Publication ;
             schema:datePublished ?pubDate ;
             ex:introduces ?model .
  ?model ex:hasRepository ?repo .
  ?repo  ex:hasRelease ?release .
  ?release ex:releaseDate ?relDate .
  FILTER(?pubDate >= "2020-01-01"^^xsd:date && ?pubDate <= "2024-12-31"^^xsd:date)
}
ORDER BY ?yearDiff
```

The query calculates the year difference between the publication and the release dates for models in the period 2020-2024. By usingFILTER it constrains the date range and ORDER BY is used for temporal analysis, which further reveals adoption lags.

CQ2: Fastest Adapted AI Frameworks in Industry

```
PREFIX ex: <http://example.org/ai-research-gap#>

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?framework (COUNT(DISTINCT ?repo) AS ?usageCount)

WHERE {

  ?model ex:usesFramework ?framework ;

         ex:hasRepository ?repo .

  OPTIONAL { ?repo ex:stars ?stars }

  FILTER(!BOUND(?stars) || ?stars > 100)

}

GROUP BY ?framework

HAVING (COUNT(DISTINCT ?repo) > 5)

ORDER BY DESC(?usageCount)
```

CQ3: Features Setting Groundbreaking Models Apart

```
PREFIX ex: <http://example.org/ai-research-gap#>

SELECT ?model ?architecture ?bench ?score ?metric
WHERE {
  ?br a ex:BenchmarkResult ;
      ex:ofModel ?model ;
      ex:forBenchmark ?bench ;
      ex:score ?score ;
      ex:metricName ?metric .
  ?model ex:implementsArchitecture ?architecture .
  FILTER(?score > 90)   # your score example uses 0–100 scale
}
ORDER BY DESC(?score)
```

The last query identifies models with benchmark scores above 90, linking to features, using FILTER to isolate high-performing models, thereby providing insights into innovation drivers.

These queries leverage the knowledge graph's 16,664 triples to offer actionable data for AI/ML engineers, companies, and researchers, with implemented advanced SPARQL features enhancing analytical depth.

8 Data Science pipeline

The data science pipeline Jupyter notebook (implementation.ipynb) processes the knowledge graph of 16, 664 triples from Papers with Code, GitHub, DBpedia, and Wikidata in "instances.ttl". The KG is read in rdflib, SPARQL queries are executed (as in Task 7), and results transformed into Pandas dataframes for specialized analysis, fulfilling the goal for numerically estimating the AI research-to-implementation gap. The pipeline responds to competency questions (CQ1-3) as follows:

CQ1 (Time lag): Compares time difference between schema:publicationDate (research works) and schema:datePublished (adoption) through dataframe operations, maintaining differences in trend analysis (6-12 month averages, for instance).

CQ2 (Framework adoption): Sums up framework usage counts (e.g., PyTorch, TensorFlow) into a dataframe, filtered by GitHub stars, to determine industry vs. academia adoption rates.

CQ3 (Features of breakthrough models): Places benchmark scores and features into a dataframe, highlighting top performers and characteristic features.

The commented pipeline, is reproducible and relies on the ontology in "final_1.ttl" for semantic structure.

Optimization Competency Questions
CQ1: Examination of time lags (e.g., 6-12 months) in dataframes guides engineers and companies on timescales for uptake in research and investment planning.
CQ2: Industry patterns in aggregated data for the framework (e.g., PyTorch leading) guide investment by indicating industry trends.
CQ3: Benchmark and feature dataframes (e.g., score 95 with 3 features) guide researchers in identifying innovation markers in curricular development.

9 Contributions and Justification

The tasks were divided effectively amongst the group's members. Dylan took charge of Task 7 (Relevant SPARQL queries) and Task 8 (Data Science pipeline) to perform adequate data analysis, and played a minor role with regards to Task 3 (Design of Ontology). Anastasia took charge of task 3 (Design of Ontology) and task 4 (Data Integration and conversion) while Nia contributed on task 1 and tasks 5 and 6 (The knowledge graph produced and Meaningful Inferences) effectively. Kareem Tujjar handled Task 2 (External sources identified), Task 9 (Contributions and Justification), and Task 10 (Usage of Generative AI). The group's coordination and team work helped with finishing the tasks on time and efficiently.

10 Usage of (Generative) AI

a) Ontology Alignment:

We asked Claude (Anthropic) to help produce the initial proposal for mapping our custom classes (like Model, Framework) to their Schema.org and FOAF equivalents. LMM proposed candidates similar to schema:SoftwareApplication for Framework and foaf:Person for Person. All proposals were, nonetheless, manually validated taking into account official ontology documentation (schema.org, FOAF spec) and examining inference in Protege. We dismissed mappings that resulted in logic problems (for example, the initial schema:Product mapping for Model, which we replaced with a custom class based on schema:CreativeWork because of domain concerns).

b) Text Correction:

ChatGPT was employed to review Section 3 (Design of Ontology) and Section 6 (Meaningful Inferences) for grammar and clarity. All changes have been checked manually to retain technical accuracy. We declare that we did not use (Generative) AI for other purposes related to this project and report (coding, visualisation, ontology construction).

References

Ontologies & Vocabularies

Brickley, D., & Miller, L. (2014). *FOAF Vocabulary Specification 0.99*. Retrieved from http://xmlns.com/foaf/spec/

Guha, R. V., Brickley, D., & Macbeth, S. (2016). Schema.org: Evolution of structured data on the web. *Communications of the ACM, 59*(2), 44-51. https://doi.org/10.1145/2844544

Data Sources

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. (2007). DBpedia: A nucleus for a web of open data. In *The Semantic Web* (pp. 722-735). Springer. https://doi.org/10.1007/978-3-540-76298-0_52

GitHub Archive. (2025). *GitHub Archive*. Retrieved October 23, 2025, from https://www.gharchive.org/

Papers with Code. (2025). *Papers with Code - Machine Learning Papers with Code*. Retrieved October 23, 2025, from https://paperswithcode.com/

Vrandečić, D., & Krötzsch, M. (2014). Wikidata: A free collaborative knowledgebase. *Communications of the ACM, 57*(10), 78-85. https://doi.org/10.1145/2629489

Ontology Methodology

Noy, N. F., & McGuinness, D. L. (2001). *Ontology development 101: A guide to creating your first ontology* (Stanford Knowledge Systems Laboratory Technical Report KSL-01-05). Stanford University.

Semantic Web Standards

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., & Rudolph, S. (2012). *OWL 2 Web Ontology Language Primer (Second Edition)*. W3C Recommendation. Retrieved from https://www.w3.org/TR/owl2-primer/