Graph-Based Patent Mining for Mechanical Designs

Manal Helal*
School of Physics, Engineering and Computer Science
Hertfordshire University
Hatfield, UK
mhelal@ieee.org
0000-0002-7515-2071

Mohammed Helal
Computer Engineering Department
Arab Academy for Science, Technology & Maritime Transport
Cairo, Egypt
info@hits-au.com

***Abstract*—Patents represent a rich source of design innovations, prompting the application of different technologies. Machine learning, text and data mining, similarity scoring, and evolving ontology methods are among the various approaches applied in the literature. This study introduces a schema-free graph data modelling of Functional Analysis Diagrams (FAD) extracted from Patents and their associated Auto-CAD models. It aims to represent mechanical design patents semantically. The schema-free graph model allows for a flexible evolving ontology of known geometries, interactions, and functions. This evolution enables comprehensive queries and ensures efficient storage that is compatible with visualisation libraries. The developed PatMine SolidWorks Add-in ©streamlines CAD design comparison with stored patents' FAD annotations by highlighting shared concepts. It also enables comprehensive full-text and semantic search queries, similarity scoring, and efficient storage compatible with visualisation libraries. The extraction of functional analysis and geometric features empowers the patent database with capabilities for seamless integration with graph analytics and machine-learning approaches for future endeavours.**

***Index Terms*—Patent Mining; Semantic Analysis; Functional Analysis Diagrams; Graph Data Modelling; Visualisation; Similarity Scoring; Big Data Analytics; Machine Learning; Artificial Intelligence**

I. Introduction

A significant challenge in big data management is managing patent data, including initial filings, applications, storage, and archiving. Automated mining techniques are necessary for knowledge discovery and understanding of prior art. Patent mining is unique compared to other data and text mining methods. Document classification, reference searches, and innovation-based decision-making are all part of Patent mining's distinctive approach to pre-filing. Following filing, ongoing activities involve the upkeep of the accepted patents dataset and its representation to assess potential infringement against existing patents. Analytical endeavours within patent mining encompass a range of tasks. These include classification, retrieval, visualisation, and valuation, which cover innovation scoring and infringement determination, as detailed in [1]. Addressing these challenges effectively requires a multifaceted approach. This approach leverages various technologies and methodologies to uncover and analyse the information in patent documents.

This study introduces a novel methodology that integrates ontology and semantic databases within a graph data model to represent the Function Analysis Diagram (FAD) extracted from patents, subsequently stored in a comprehensive patent knowledge base. This approach offers enhanced performance, visualisation, similarity scoring, other analysis insights, and greater adaptability to emerging technologies than conventional methods. The objectives of this research encompass establishing a semantic database utilising a suitable data model, visually presenting patent contents through informative diagrams, and facilitating the scoring of similarity and the other tasks in patent analytics.

The paper begins with a literature review, followed by a section on methods that outline the proposed graph modelling approach for representing FADs extracted from mechanical engineering patents. Subsequently, the results section details user interface (UI) specifications, search methodologies, similarity scoring mechanisms, and visualisation strategies. The paper concludes with a summary of its findings and remarks encapsulating the critical insights obtained.

II. Literature Review

A patent file encompasses structured data, such as patent numbers, inventor details, filing, issue dates, and assignees. They also include various forms of unstructured content, including abstracts, claims, descriptions, specifications, and illustrations. Despite possessing more structure than web pages, patents present challenges for effective full-text keyword searches due to their length, technical terminology, legal jargon, and multilingual content. This inherent complexity makes patent analysis challenging, even for seasoned experts [1]. Systematic approaches in the literature to extract knowledge from engineering patent documents include the Theory of Constraints (TOC), Quality Function Deployment (QFD), Axiomatic Design (AD), and Value Analysis (VA). For instance, the TOC method enhances keyword selection by incorporating physical analysis and leveraging an energy converters database and technical system evolution trends [2].

The evolution of data models, progressing from file systems to relational database management systems (RDBMS), has seen the emergence of various alternatives such as object-oriented (OO), XML, RDF, OWL, NoSQL databases, and Graph Databases. Alongside these advancements, several graph query languages like SPARQL, GraphQL, Gremlin, and OpenCypher have emerged, each offering distinct strengths

and limitations [3]. Graph data modelling is particularly advantageous in scenarios where interconnections or topology play a crucial role, especially when dealing with datasets with unknown or dynamic schema or where not all values are present.

Ontology designs are vital in patent informatics research, employing controlled vocabularies, taxonomies, and semantic relationships for systematic storage in semantic databases. This facilitates subsequent querying and analysis tasks. Ontology building for CAD/CAM models involves intensive data handling and often necessitates using controlled taxonomies in the initial design. Various previous work proposed CAD ontology building approaches [25]–[27]. The need to formalise ontology-building approaches for consistent migration between products is identified in an earlier work proposing various standardisation approaches [28], [29].

The Function Analysis Diagram (FAD) method simplifies form-dependent functional descriptions through intuitive graphical notation, effectively capturing intricate functional interactions [6]. However, employing structured semantic databases for FAD models presents challenges such as limited visualisation options, performance issues with relational databases, and difficulties quantifying similarity between complex FAD models. To address these challenges, utilising graph databases for semantic FAD models offers notable benefits. Graph databases enable evolving ontology-building taxonomy as more products are annotated. The most suitable taxonomy can be reached by consensus using automated voting if crowdsourced, or expert annotations and corrections. Product abstract similarity can be computed using network alignment algorithms to align patents' graph models, facilitating accurate similarity score calculation. Alignment allows for comparisons at a conceptual level with variable penalties for insertions, modifications, and deletions of geometries, interactions and/or functions, enhancing the effectiveness of patent analysis methodologies.

III. Graph Modelling of FAD Ontology

This paper introduces the PatMine prototype ©, which employs graph modelling of FAD models using Neo4j. The presented prototype illustrates the interactions among fundamental geometric features and their functional behaviours, referred to as Functional Geometry Interactions (FGIs). FGIs also group the geometric features according to their topological relationships. The design's innovative operational principles, termed Key-FGIs, are incorporated into the FAD model. The FAD model is organised into three tiers. At the abstract level, functional design claims are recorded in natural language. Geometric features are outlined at the following level, and composed of detailed subcomponents. The third level illustrates how functions are realised through interactions between geometric features of the system.

FAD intuitively facilitates graph modelling using a set of FGIs as edges between geometric features serving as basic node types. Additional information can attribute nodes and edges as properties. For visualisation, carefully choosing labels and properties simplifies complexity, ensuring that only necessary node types are displayed in graphical representations. However, queries for statistical analysis can reveal the hidden complexity in property values. FAD annotates a design outlined in a patent document or a fresh CAD model to be compared with other FAD models stored in the proposed database. Relevant ontology is uploaded to the system as prefilled user selection options for the various properties, such as geometry types, interaction types, function types and others.

Patents are depicted as nodes in the graphical representation, with multiple products represented by several subgraphs. Each product contains nodes representing geometric features connected by FGI edges. Geometric feature levels of abstraction utilise an is-A relationship, where, for example, a level 5 square object is a sub-type of the level 4 polygon super-type. FGIs' actions, as edge attributes, describe the interaction between two geometries and their contribution to accomplishing the claimed function. Clusters of FGIs execute sequential steps to fulfil a function, with explicit instructions for the sequence of steps. For instance, two sequential steps are captured in opening a can by two FGIs: first, a latch geometry applies pressure to the can cover geometry, followed by the cover geometry lifting and detaching from the can body geometry. The following Cypher query depicts this example:

```
latch-[press]→cover; cover-[separates]→can body
```

All FGI edges that contribute to executing a function's steps are represented as separate sub-graphs linked to the design root node with distinct function ID properties. When querying for a function, the system generates a list of geometries connected with FGIs associated with the function's ID or name. New behaviours for a function are defined as attribute values. For instance, a radiator can serve for both heating and cooling purposes. This functionality can be modelled as a "change in environment temperature" function, with two behaviours: "heating" and "cooling." Each patent is represented by a graph with edges connecting to the product(s) it contains. Each product is associated with the geometries it includes, and these geometries form FGI relationships. These relationships encompass properties that detail the function and behaviour they belong to and the action they execute. This is expressed in Cypher Query as:

```
match (p: patent {Patent_Number: "PatentUniqueIdentifier"})
optional match (p)-[:hasProduct]->(pr: product)
optional match (pr)-[:hasClaim]->(c)
optional match (pr)-[:hasGeometry]->(g1)
optional match (g1)-[fr:hasFGI]->(g2)
return p, pr, c, g1, fr, g2
```

Products, claims, and geometries can all be constrained by entity constraints. In the following Cypher query, the pattern for level two FAD is demonstrated after the initial matching

process:

```
match (p:patent {Patent_Number: "PatentUniqueIdentifier"})
match (pr:product {Product_ID: "Product_ID"})
create (g:geometry {name: "Geometric_Feature_Name",
  Geometric_ID: "Geometric_Feature_ID",
  PatMine_type: "Geometric_Feature_Type"})
create (pr)-[:hasGeometry]->(g)
return p, pr, g
```

The following Cypher statement defines the patent/design and, optionally, the product to which an FGI edge is added to connect existing geometries:

```
match (p:patent {Patent_Number: "PatentUniqueIdentifier"})
match (pr:product {Product_ID: "Product_ID"})
match (g1:geometry  {name: "geomName1",
  Geometric_ID: "geomID1"})
match (g2:geometry  {name: "geomName2",
  Geometric_ID: "geomID2"})
create (g1)-[:hasFGI {Function_IDs:
  "Function_ID_List", action: "actionType"}] >(g2)
return p, pr, g1, g2
```

The Function ID property on FGI relations allows us to find all third-level FAD elements that comprise a function's structure. Here's a Cypher statement for performing this search:

```
match (g1)-[r1:hasFGI]->(g2)
where "queryFunctionID" in r1.Function_IDs
match (p)-[:hasProduct]->(pr)
match (pr)-[:hasGeometry]->(g1)
return p, pr, g1, r1, g2
```

For a completed FAD model for a new design, a full-text search using its keywords retrieves all matching FADs in PatMine database. The retrieved models can be augmented with those retrieved from text search over the pdf files of the patents. Full-text search is enabled by utilising Lucene/Solr or Elastic Search plugins for Neo4j. It can also be implemented by matching keywords to values in the designs' properties:

```
match path = (p: patent) -[:hasProduct]->(pr: product)
  -[:hasGeometry]->(g1) -[fr:hasFGI]->(g2)
  with nodes(path) AS x UNWIND x AS nodeLinks with nodeLinks,
  [prop in keys(nodeLinks)
WHERE nodeLinks[prop] in ["queryKeyword1", " queryKeyword2", ...]] as matchRank
where size(matchRank) > 0
return nodeLinks, matchRank
```

The provided query retrieves paths from every patent in the database to every FGI it encompasses. The retrieved paths are then evaluated based on matched keywords, and the cumulative score is computed for each patent. Other technique can be implemented, such as query expansion with synonyms from the PatMine ontology or WordNed. The merged results generated can be implemented semantically relevant patents to the query design, ranked according to keywords used in the FAD models, and should be semantically associated to enhance the search results' relevance. Below is an example of a semantic query at FAD level two exploring relevant FGIs between geometries:

```
match (p)-[:hasProduct]->(pr)
match (pr)-[:hasGeometry]->(g1)
match (g1)-[r1:hasFGI]->(g2)
where g1.PatMine_TYPE =~ ".queryPatMineType1."
  and g2.PatMine_TYPE =~ ".queryPatMineType2."
  and r1.action = "queryAction"
  and filter(funID IN r1.Function_IDs WHERE funID = "queryFunctionID")
return p, pr, g1, r1, g2, count(r1)
as MatchRank2
```

Function names' synonyms should be incorporated into the "where" clause:

```
where r1.Function_Name in
["SynSet1"," SynSet2", ...]
```

IV. Results

Visual Studio C# APIs with SolidWorks add-in is the development environment used for front-end Patent Mining (PatMine) ©tool interfacing with the Neo4j database as the back-end. Designed to maintain the schema-free nature of the Neo4j, the tool serialises Neo4j query outcomes into object types that capture node properties. The query results are the stored patents or the previously stored new designs objects containing products, claims, geometries, and FGIs. The node's primary type is determined by its first label.

Data about patents or new designs can be uploaded from formatted Excel sheets, facilitating the bulk upload of large-volume annotations. Future developments may entail decoupling PatMine from SolidWorks and enabling integration with other CAD tools. The tool offers various functionalities, outlined as follows:

- **FAD Model components definition UI**. Divided into three tabs:
  - Design and product names along with provided claims.
  - Geometric features list contained within the design.
  - FGIs List connecting geometries. Fig. 1 illustrates a corkscrew example with annotations and visualisations alongside the SolidWorks CAD model.
- **Searching and browsing FAD models**. Fig. 2 depicts the three search options and their results illustrated in Fig. 3 for the cork extracting apparatus. These resulting patents are graphically displayed with zooming functionality, allowing for a thorough investigation of interactions. Navigation buttons facilitate movement between matched patents in the database. When a user does not define query values, the query results will contain all database-modelled patents.
  - Semantic search queries from keywords in the properties' values such as title, product, function, action, and geometry nodes.
  - Dynamic full-text search from FAD model keywords.
  - Typing cypher statements by experienced users.
- **FAD models comparisons**. The comparison results are visualised through charts indicating matching scores and

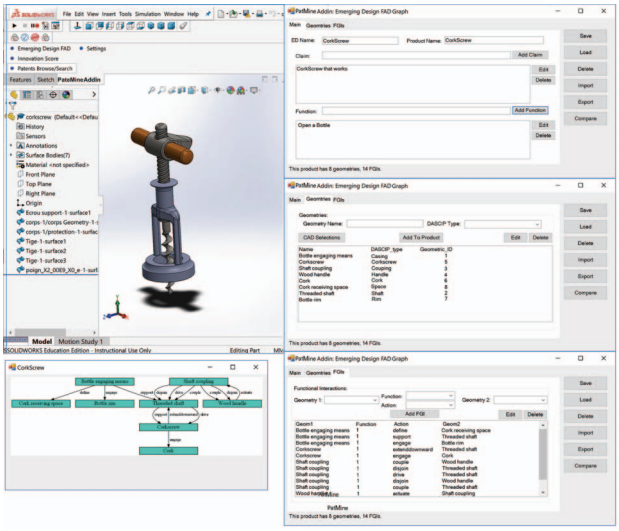

Fig. 1. Corkscrew FAD model definition and visualisation

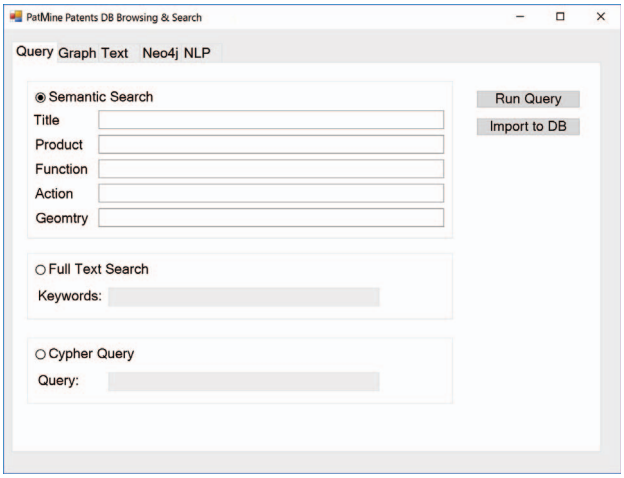

Fig. 2. Different queries' options

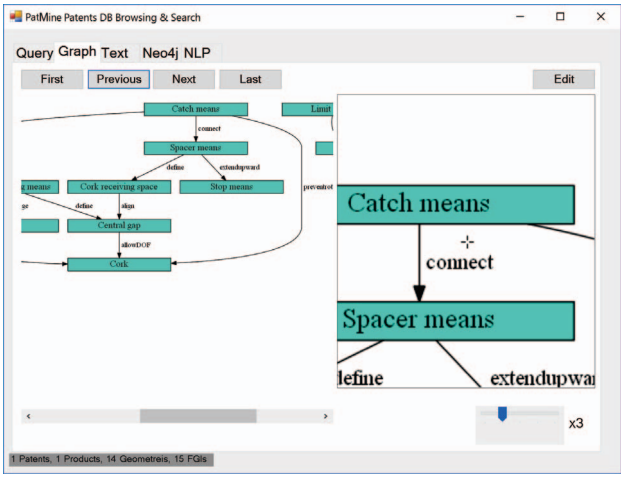

Fig. 3. Search Results

highlighting overlapping geometries and interactions. Retrieval of corresponding patent documents is facilitated, featuring highlighted FAD annotations and image thumbnails of crucial overlapping drawings. In the scoring mechanism, similar geometric features hold a weight of 10 in FAD level one, while similar FGIs are assigned a weight of 20 in FAD level two. In FAD level three, overlaps among combinations of FGIs forming functions are tallied, with the highest weight of 30. The overall similarity score is computed using a weighted sum of these counts. Normalisation across all patent match scores is performed as follows:

```
Match Rank = (Match Rank { Minimum Match Rank )
        / (Maximum Match Rank { Minimum Match Rank)
```

Fig. 4 illustrates scoring outcomes for a corkscrew example, showcasing a detailed interface for exploring overlapping FAD annotations and accessing original patent documents. The ontology type of the geometry is utilised for matching instead of user-provided names, and sophisticated techniques like network alignment algorithms can be leveraged for indirect relationships and identifying similar overlapping regions, with adjustments made for less significant insertions and deletions in FAD models' graph data structures.

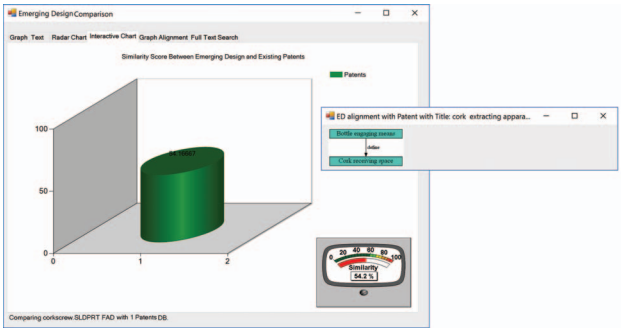

Fig. 4. Corkscrew Design similarity score of 54%, with identified overlapping interactions

- **FAD models Visualisation**. Patent documents are depicted on the highest level as graphs, linking patent citations to capture semi-structured information. While citation graphs offer limited internal details, network-based patent analysis methods, as discussed by [15], offer a broader perspective by illustrating overall relationships among patents in visual networks. However, deeper insights into patents' content and relevance are found in the unstructured information within patent documents. These details are captured in the FAD patent representation as the geometries nodes and their interaction edges doing actions in function on the detailed level. Previous studies, including those by [15]–[18], have introduced various patent maps. The proposed prototype utilises a Neo4j Server in the back end and its browser, providing an online platform for users familiar with Cypher to query

and interact with the emerging patents graph database instance. This browser offers visualisation through a force-directed graph drawing algorithm, which utilises character co-occurrence as a force within a physical simulation resembling charged particles and springs. Fig. 5 and Fig. 6 exemplify the visualisation of a corkscrew and corkscrew apparatus patent, respectively.

Fig. 5. Corkscrew Design FAD Neo4j Server Browser Visualisation

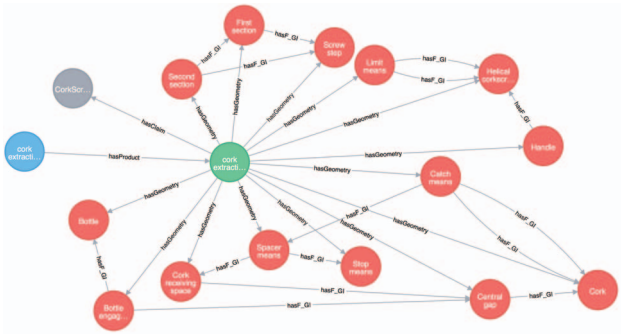

Fig. 6. Corkscrew Apparatus FAD Neo4j Server Browser Visualisation

GraphViz, a suite of tools, [22] provides another method of formal visualisation through their .Net wrapper, leveraging the dot language [23]. This process involves transforming Neo4j query results from tabular format to serialised object-oriented entities and relationships and then converting them to the dot language format to generate the required visualisation. The previously discussed corkscrew design and apparatus patent examples are visualised in the proposed prototype in Fig. 7 and Fig. 8, respectively. This visualisation proves valuable in detecting interaction clusters that may suggest one or multiple functions, geometries lacking interactions, and functions subdivided into clear sub-functions, leading to distinct sub-graphs. Functionalities like zooming in the thumbnail image of the patient's FAD are shown in Fig. 8, and other updates/deletes of the FAD model are available in the proposed prototype.

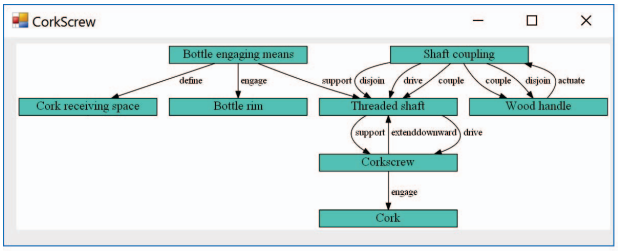

Fig. 7. Corkscrew FAD proposed formal visualisation

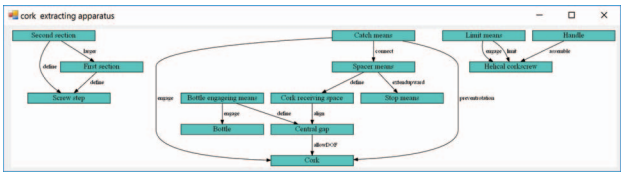

Fig. 8. Corkscrew Apparatus FAD proposed formal visualisation

- **Ontology Forming**. Users are provided with access to all environment variables and system settings for updating, enabling dynamic customisation of system behaviour and evolving ontology. Formatted Excel sheets containing multiple FAD annotations are used to populate the database. Users also have the option to add annotations individually for each patents. As the database grows, there is potential for exploring Neo4j distributed architectures on the varying cluster sizes. If made available online for crowd-sourcing, unverified annotations can be accepted after expert verification, updating the ontology taxonomy as required.

V. Conclusion

By combining graph databases with ontology building and semantic databases, we present an innovative approach to modelling both. Graph models offer several advantages for representing PatMine ontology and structured patent information. First, the schema-free nature of these databases enables the ontology to evolve with contributions from crowed-sourced patent FAD annotations, then corrected by domain experts or natural language processing models [10], [30]. The process can define new rules, entity types, relationship types, and characteristics. Connecting to other databases and knowledge bases involves adding edges between nodes in different graph structures. A previous study provided an example that imported XML metadata into a Neo4j database to establish various relationships [8]. Second, since patents are semi-structured documents, rules can be defined for parsing them and extracting structured information from unstructured text in multiple ways within a specific domain [11]. Using computer vision deep learning models, it is possible to automatically extract FAD graph structure from patents after accumulating crowd-sourced and verified FAD extractions from AutoCAD drawings as a training dataset. Third, the system's performance surpasses sequential file access provided by RDBMS or independent files. The performance of graph modelling

over RDBMS was estimated to be significantly higher [13]. For instance, a three-level relationship operates 150 times faster, while a four-level deep query demonstrates superior performance by a factor of 1000.

Fourth, graph data structures facilitate the automatic generation of FAD models from different visualisation options. As a result of representing patent information in simpler geometric components and their interactions, similarities can be quantified based on their interactions or recent development of various metrics [31]. Fifth, analytical algorithms on graphs can summarise patents and identify industry trends and common shared concepts. Designers of emerging innovations can benefit from various alarms as common concepts from the attached patents database are used to redesign using less frequently used concepts or propose completely new concepts. Lastly, leveraging the proposed prototype as a foundation for AI and machine learning research will enhance the presented prototype knowledge base, as well as the visualisation, similarity scoring, and searching functions presented by incorporating inference and logic techniques such as unification, lambda calculus, and Peirce's existential graphs [24].

REFERENCES

[1] L. Zhang, L. Li, and T. Li, "Patent Mining: A Survey," SIGKDD Explor. Newsl., vol. 16, no. 2, pp. 1-19, May 2015, doi: 10.1145/2783702.2783704.

[2] U. Y. Valverde, J.-P. Nadeau, and D. Scaravetti, "A new method for extracting knowledge from patents to inspire designers during the problem-solving phase," Journal of Engineering Design, vol. 28, no. 6, pp. 369–407, Jun. 2017, doi: 10.1080/09544828.2017.1316361.

[3] O. Panzarino, Learning cypher: write powerful and efficient queries for Neo4j with Cypher its official query language. Birmingham, U.K: Packt Pub, 2014.

[4] C. Fellbaum, Ed., WordNet: An Electronic Lexical Database. The MIT Press, 1998. doi: 10.7551/mitpress/7287.001.0001.

[5] W. Li, Y. Li, J. Chen, and C. Hou, "Product functional information based automatic patent classification: Method and experimental studies," Information Systems, vol. 67, pp. 71–82, Jul. 2017, doi: 10.1016/j.is.2017.03.007.

[6] M. Aurisicchio, R. Bracewell, and G. Armstrong, "The Function Analysis Diagram," in Volume 7: 9th International Conference on Design Education; 24th International Conference on Design Theory and Methodology, Chicago, Illinois, USA: American Society of Mechanical Engineers, Aug. 2012, pp. 849–861. doi: 10.1115/DETC2012-70944.

[7] S. Bordoloi and B. Kalita, "Designing Graph Database Models from Existing Relational Databases," IJCA, vol. 74, no. 1, pp. 25–31, Jul. 2013, doi: 10.5120/12850-9303.

[8] J. Wang, B. Evans, L. Wyborn, A. Aryani, and M. Barlow, "Graph Connections Made By RD-Switchboard Using NCI's Metadata," D-Lib Magazine, vol. 23, no. 1/2, Jan. 2017, doi: 10.1045/january2017-aryani.

[9] S. R. Bowman, J. Gauthier, A. Rastogi, R. Gupta, C. D. Manning, and C. Potts, "A Fast Unified Model for Parsing and Sentence Understanding," in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Berlin, Germany: Association for Computational Linguistics, 2016, pp. 1466–1477. doi: 10.18653/v1/P16-1139.

[10] G. Angeli, M. J. Johnson Premkumar, and C. D. Manning, "Leveraging Linguistic Structure For Open Domain Information Extraction," in Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Beijing, China: Association for Computational Linguistics, 2015, pp. 344–354. doi: 10.3115/v1/P15-1034.

[11] D. Ferrucci and A. Lally, "UIMA: an architectural approach to unstructured information processing in the corporate research environment," Nat. Lang. Eng., vol. 10, no. 3–4, pp. 327–348, Sep. 2004, doi: 10.1017/S1351324904003523.

[12] D. M. Jessop, "Information extraction from chemical patents, "Degree of Doctor of Philosophy Thesis, Apollo-University Of Cambridge Repository, 2011. [Online]. Available: https://www.repository.cam.ac.uk/handle/1810/238302

[13] H. R. Vyawahare and P. P. Karde, "An overview on graph database model," International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE), vol. 3, no. 8, pp. 7455–7457, 2015.

[14] M.-H. Huang, L.-Y. Chiang, and D.-Z. Chen, "Constructing a patent citation map using bibliographic coupling: A study of Taiwan's high-tech companies," Scientometrics, vol. 58, no. 3, pp. 489–506, Nov. 2003, doi: 10.1023/B:SCIE.0000006876.29052.bf.

[15] B. Yoon and Y. Park, "A text-mining-based patent network: Analytical tool for high-technology trend," The Journal of High Technology Management Research, vol. 15, no. 1, pp. 37–50, Feb. 2004, doi: 10.1016/j.hitech.2003.09.003.

[16] H. Uchida, A. Mano, and T. Yukawa, "Patent Map Generation using Concept-based Vector Space Model," in Working Notes of NTCIR-4, Tokyo, Japan: National Institute of Informatics, Jun. 2004, pp. 2–4.

[17] B. Yoon, C. Yoon, and Y. Park, "On the development and application of a self–organising feature map–based patent map," R & D Management, vol. 32, no. 4, pp. 291–300, Sep. 2002, doi: 10.1111/1467-9310.00261.

[18] S. Lee, B. Yoon, and Y. Park, "An approach to discovering new technology opportunities: Keyword-based patent map approach," Technovation, vol. 29, no. 6–7, pp. 481–497, Jun. 2009, doi: 10.1016/j.technovation.2008.10.006.

[19] R. Hull and R. King, "Semantic database modeling: survey, applications, and research issues," ACM Comput. Surv., vol. 19, no. 3, pp. 201–260, Sep. 1987, doi: 10.1145/45072.45073.

[20] R. Tamassia, Ed., Handbook of graph drawing and visualisation, First issued in paperback. in Discrete mathematics and its applications. Boca Raton London New York: CRC Press, 2016.

[21] M. Consens and A. Mendelzon, "Hy+: a Hygraph-based query and visualisation system," SIGMOD Rec., vol. 22, no. 2, pp. 511–516, Jun. 1993, doi: 10.1145/170036.171537.

[22] E. R. Gansner and S. C. North, "An open graph visualisation system and its applications to software engineering," Softw: Pract. Exper., vol. 30, no. 11, pp. 1203–1233, Sep. 2000, doi: 10.1002/1097-024X(200009)30:11¡1203::AID-SPE338¿3.0.CO;2-N.

[23] Emden R. Gansner, Eleftherios Koutsofios, and Stephen North. 2002. "Drawing Graphs with Dot." Murray Hill, NJ: AT&T Bell Laboratories. Available: http://www.research.att.com/sw/tools/graphviz/dotguide.pdf

[24] J. F. Sowa, "Conceptual graphs as a universal knowledge representation," Computers & Mathematics with Applications, vol. 23, no. 2, pp. 75–93, 1992, doi: https://doi.org/10.1016/0898-1221(92)90137-7.

[25] M. Huang, S. Sui, W. Mou, S. Zhang, and W. Cao, 'Three-dimensional CAD Model Retrieval Algorithm Based on Ontology', Procedia CIRP, vol. 56, pp. 590–593, 2016, doi: 10.1016/j.procir.2016.10.116.

[26] C. Benavides, I. García, H. Alaiz, and L. Quesada, 'An ontology-based approach to knowledge representation for Computer-Aided Control System Design', Data & Knowledge Engineering, vol. 118, pp. 107–125, Nov. 2018, doi: 10.1016/j.datak.2018.10.002.

[27] S. Nzetchou, A. Durupt, S. Remy, and B. Eynard, 'Semantic enrichment approach for low-level CAD models managed in PLM context: Literature review and research prospect', Computers in Industry, vol. 135, p. 103575, Feb. 2022, doi: 10.1016/j.compind.2021.103575.

[28] Y.-C. Lee, C. M. Eastman, and W. Solihin, 'An ontology-based approach for developing data exchange requirements and model views of building information modeling', Advanced Engineering Informatics, vol. 30, no. 3, pp. 354–367, Aug. 2016, doi: 10.1016/j.aei.2016.04.008.

[29] Y. Liu, 'Consistency Checking Based on Ontology of Design Information', AMM, vol. 438–439, pp. 1992–1997, Oct. 2013, doi: 10.4028/www.scientific.net/AMM.438-439.1992.

[30] M. Yin, L. Tang, C. Webster, X. Yi, H. Ying, and Y. Wen, 'A deep natural language processing-based method for ontology learning of project-specific properties from building information models', Computer aided Civil Eng, vol. 39, no. 1, pp. 20–45, Jan. 2024, doi: 10.1111/mice.13013.

[31] M. Weber and R. Anderl, 'Ontology-Based Calculation of Complexity Metrics for Components in CAD Systems', in Uncertainty in Mechanical Engineering, P. F. Pelz and P. Groche, Eds., in Lecture Notes in Mechanical Engineering. , Cham: Springer International Publishing, 2021, pp. 3–11. doi: 10.1007/978-3-030-77256-7_1.