Integrating and Navigating Decision-Related Knowledge Using Decision Knowledge Graph

Jia Hao
School of Mechanical Engineering
Beijing Institute of Technology, Beijing, P.R. China
No. 5 Zhongguancun South Street, Haidian District, Beijing, China 100081
haojia632@bit.edu.cn

Lei Zhao
School of Mechanical Engineering
Beijing Institute of Technology, Beijing, P.R. China
No. 5 Zhongguancun South Street, Haidian District, Beijing, China 100081
zhao_lei@bit.edu.cn

Jelena Milisavljevic-Syed
Systems Realization Laboratory,
Division of Industrial Design
School of Engineering
The University of Liverpool
j.milisavljevic-syed@liverpool.ac.uk

Zhenjun Ming[1]
School of Mechanical Engineering
Beijing Institute of Technology, Beijing, P.R. China
No. 5 Zhongguancun South Street, Haidian District, Beijing, China 100081
zhenjun.ming@bit.edu.cn

[1] Corresponding Author

ABSTRACT

Designers are usually facing a problem of finding information from a huge amount of unstructured textual documents in order to prepare for a decision to be made. The major challenge is that knowledge embedded in the textual documents are difficult to search at a semantic level and therefore not ready to support decisions in a timely manner. To address this challenge, in this paper we propose a knowledge-graph-based method for integrating and navigating decision-related knowledge in engineering design. The presented method is based on a meta-model of decision knowledge graph (mDKG) that is grounded in the compromise Decision Support Problem (cDSP) construct which is used by designers as a means to formulate design decisions linguistically and mathematically. Based on the mDKG, we propose a procedure for automatically converting word-based cDSPs to knowledge graph through natural language processing, and a procedure for rapidly and accurately navigating decision-related knowledge through divergence and convergence processes. The knowledge-graph-based method is verified using the textual data from the supply chain design domain. Results show that our method has better performance than the conventional keyword-based searching method in terms of both effectiveness and efficiency in finding the target knowledge.

Keywords: Design; Decision Support; Knowledge Graph; Searching; Navigation

1. Introduction

Engineering design is increasingly recognized as a decision-making process [1-3]. Mistree et al. [4] point out that the principal role of a designer is to make decisions. Providing relevant decision support is critical for augmenting designers' decision-making ability and thus speeding up the design process and generating quality designs. In engineering design, how designers make decisions is typically summarized as a three-step iterative process, as shown in Figure 1.

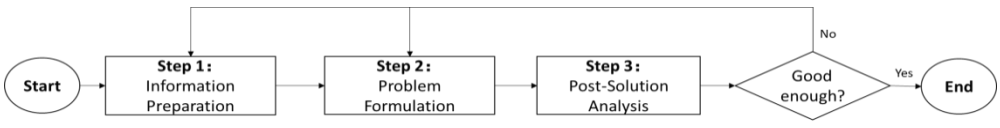

Figure 1 A General Process for Decision Making in Design

The three steps are as follows:

- *Step 1*: Gathering relevant information, the information comes from stakeholders

- *Step 2*: Formulating a decision problem using the gathered information, we transform the information into structural form
- *Step 3*: Performing post-solution analysis to decide whether or not the solution is good enough, if not, refine *Steps 1* and *2*.

Step 1 deals with the preparation of information that is fundamental for decision formulation in Step 2 and post-solution analysis in Step 3. Information preparation is important due to the fact that designers need certain amount of domain-specific information or knowledge to understand to problem before it is formulated. Li et al. [5] summarize that designers searching information mainly for four reasons: 1) acting as "memory extension" for individual engineers and enabling information sharing among them, 2) exploring design concept alternatives during the early design stage, 3) learning from the original design and understanding the rationale behind the decisions made, and 4) searching past designs for knowledge reuse when working on a similar product or problem. Despite its importance, information preparation is also a challenging task. One of the challenges is anchored in the fact that most of the engineering information sources are those unstructured, informal, text-based documents [5], which are difficult to analyze and therefore not ready to support decisions. For example, the knowledge about the design of a gearbox may be documented in technical reports or research papers. These textual documents usually have different formalities and are filled with domain-specific jargon, abbreviations, and shorthand. The inconsistency makes understanding the knowledge archived in these documents a time-consuming task. Another challenge is anchored in the difficulty of locating or navigating the wanted or interested information from a huge amount background information. For example, a designer typically needs to go through the whole document just to identify the design variables that he/she needs to consider in the design of a particular gear in a gearbox. Traditional keyword-based searching methods [6] are the commonly used means to navigate the wanted information. However, due to the lack of incorporation of contextual information in the search engine, keyword-based searching methods usually lead to either too much or irrelevant results for the request or are not in a form that designers can navigate and explore [5]. There is a need of extracting the decision-related knowledge from the textual documents and representing is in a structured manner that facilitates navigation.

In order to address this need, in this paper we propose a knowledge-graph-based method for structuring and navigating decision-related knowledge in the design of engineered systems. Knowledge graph, defined as the network of entities and their semantic relationships [7], are widely used for representing knowledge in different domains. Due to its root in graph theory, the advantages of knowledge graph include representing more complex relationships (compared with traditional relational databases), navigating relevant information in a quicker way, and enabling users to see the connection of different pieces of information in a meaningful way. In the context of decision support in engineering design, it is promising to use knowledge graph as a means to representing text-based knowledge and navigating it for decision support in design. The knowledge-graph-based

approach in this paper is implemented as one of the functions of a Cloud-Based Decision Support (CBDS) platform that is currently under development. In CBDS, the domain-dependent, -independent, and integrated knowledge are stored as services in a service pool where users can retrieve and customize for their specific needs of decision support. Knowledge graph is serving as an intermediary to convert unstructured textual document to formal structured decision support knowledge (or service), and a bridge to deliver knowledge to users in CBDS.

The rest of the paper is structured as follows. In Section 2 there is an illustration of some related research works and the research gaps are also pointed out. In Section 3 the details of the proposed method are described. In Section 4 a case study is detailed to verify this method. In Section 5 the results are analyzed and discussed while in Section 6 there is a summary of this work and some possible future works are presented.

2.Frame of Reference

2.1 Decision-Based Design

Decision-Based Design (DBD) is a paradigm for designing and creating design methods rooted in the notion that decision-making is fundamental to design [4]. The notion of DBD has motivated many research thrusts such as multi-attribute decision analysis [8], preference modeling [9], risk analysis [10], uncertainty management [11], etc. The other implementation of DBD is the Decision Support Problem Technique (DSPT) proposed by Mistree et al. [12] with an assumption that models are incomplete and inaccurate, and designers seek satisficing (good enough) solutions. DSPT is anchored in the notion that there are only two primary types of decisions, namely, selection and compromise, and that any complex design can be modeled as a network of selection and compromise decisions. Selection involves a choice among a number of possibilities considering a number of measures of merit of attributes. Compromise involves determining the “right” values of design variables to describe the best satisficing solution with respect to constraints and multiple goals. The selection Decision Support Problem (sDSP) [13] and compromise Decision Support Problem (cDSP) [14] are the constructs for formulating the selection and compromise decisions respectively. Due to their domain-independent nature, these constructs are applied in the design of different products or systems including ship [15], aircraft [16], structure material [17], thermal system [18], etc.

Given Alternative to be improved Assumptions used to model the domain Some helpful relations The system parameter The constraints and goals for the system **Find** The values of the system variables Values of deviation variables **Satisfy** System constraints System goals (normalized) Bounds on System Variables **Minimize** Deviation Function	**Given** n number of system variables p+q number of system constraints p equality constraint q inequality constraints m number of system goals $g_i(X)$ system constraint function $G_i(X) = C_i(X) - D_i(X)$ $f_k(X)$ function of deviation variables **Find** X_i System variables i = 1, …, n d_i^+, d_i^- Deviation Variables i = 1, …, 2m **Satisfy** *System constraints* $g_i(X) = 0$ i = 1, …, p $g_i(X) \geq 0$ i=p+1, …, p+q *System goals (linear, nonlinear)* $A_i(X) + d_i^- - d_i^+ = G_i$ i = 1, …, m *Bounds* $X_j^{min} \leq X_j \leq X_j^{max}$ i = 1, …, n $d_i^+, d_i^- \geq 0$; $d_i^+ \bullet d_i^- = 0$ **Minimize** Deviation Function: Archimedean $Z = \Sigma^m_{i=1} w_i (d_i^+, d_i^-)$ Deviation Function: Preemptive $Z = [f_1(d_i^+, d_i^-), \ldots, f_k(d_i^+, d_i^-)]$ i = 1, …, m
(a) cDSP Word Formulation	**(b) cDSP Mathematical Formulation**

Figure 2 cDSP word-based and mathematical Formulations

In this paper, we adopt DSPT as our implementation of DBD and propose to use cDSP as our construct for modeling decisions in design (a sDSP can be converted to a cDSP with binary variables). The word and mathematical formulations of the cDSP are shown in Figure 2-a and 2-b respectively. Nellippallil et al. [19] identify a three-step process for formulating and solving design problems using the cDSP construct.

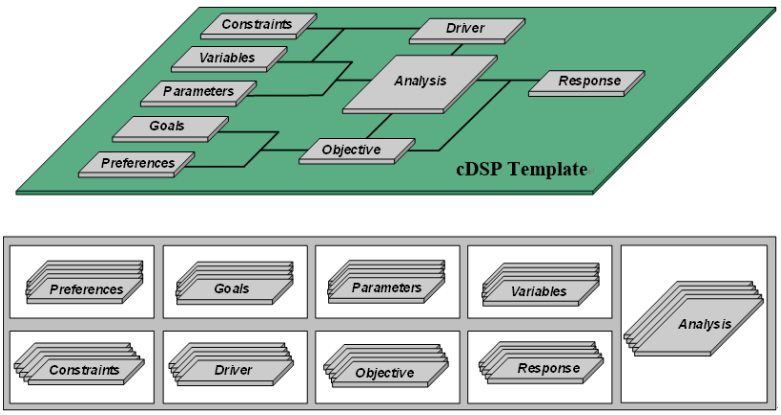

Figure 3 The cDSP Template

Representing the decision-related information in a modular and reusable manner is a feasible way to tackle the challenge of information preparation (because this will save time for re-building the model from scratch). In order to enable the re-usability of the decision-related information, Panchal et al. [20] propose the cDSP template for representing the compromise decision-related information, as shown in Figure 3. The cDSP template is comprised of two parts, namely, the "wiring board" which stands for the procedural (problem-independent) knowledge of how the decision (or interaction) is executed, and the modules which stand for the declarative (problem-dependent) knowledge of what the available resources are. The separation of the "wiring board" and the modules allows both to be reused in similar problems or across problems. Ming et al. [21] extend the idea of cDSP template to further represent it using frame-based ontology in order to facilitate designers populating knowledge instances in a knowledge base for future reuse. Based on the cDSP template and the cDSP ontology, we are now developing a Cloud-Based Decision Support (CBDS) platform. In CBDS, all the cDSP templates are stored in a service pool on the cloud and are available for customization and application in solving design problems. The issues that we face in developing the cDSP template service pool include 1) how to convert text-based cDSP formulations to cDSP template instances, and 2) how to locate the wanted modules or template instances from a huge amount of data, and 3) can we find new knowledge given a number of cDSPs. Since most of the existing cDSPs are formulated in text-based documents such as research papers and technical papers, in order to populate data in the cDSP template service pool users currently have to manually extract the modules from existing documents and input them to the template, which is time-consuming. And when the number of cDSP template instances increases, it will be difficult for users to search for the interested modules. There is a need of a method for automatically converting the text-based cDSP formulations to structured cDSP templates in the CBDS platform and navigating the cDSP modules based on user-specific requests.

2.2 Knowledge Graph in Design

The concept of Knowledge Graph (KG) is first proposed by Google in 2012 as an enhancement of

their search engine with semantics. Ehrlinger et al. [22] review the existing literature about KG and present several selected definitions of KG, for example, "knowledge graphs are large networks of entities, their semantic types, properties, and relationships between entities [7]," and "knowledge graphs mainly describe real world entities and their interrelations, organized in a graph [23]." Most of the KG definitions of are derived from applications. In this paper we adopt a more essential and structural definition of KG from Ringsquandl et al.[24], namely, a KG is a collection of triples `<entity, predicate, entity>` that form a directed graph where nodes are entities and edges are labeled with binary predicates (relations). Nodes represent entities and edges represent binary relations (predicates). Figure 4 presents an example of KG. Nodes are labelled by names of entities (e.g., Bob, Alice, Person, etc.) and edges by names of relations (e.g., is-a, is-a-friend-of, etc.) between the entities. Computationally, this KG is represented as a collection of triples as [`<Bob, is-a, Person >`; `<Bob, is-a-friend-of, Alice >`; …]. Due to the advantage in structuralizing domain knowledge and capturing the semantic relationships (linkage) among entities, KGs are used in many applications including Web search [25], recommendation [26], navigation [27], question answering [28], machine learning [29], data integration [30], entity disambiguation and linking [31]. In this section, we first review the applications of KGs in the domain of engineering design, then the paper discusses some existing methods for KG construction, finally we identify our approach to the construction and application of KG for supporting decision making in design.

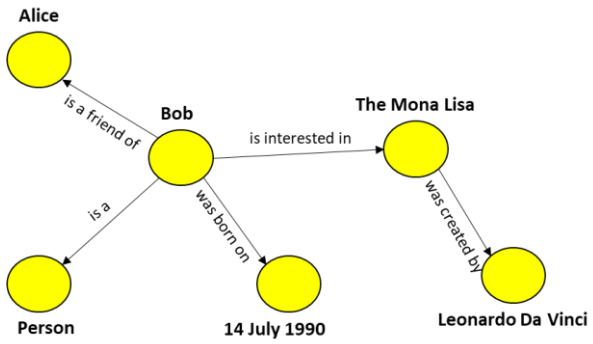

Figure 4 A Simple Example of Knowledge Graph[45]

One of the typical applications of KGs in engineering design is to enable sharing and reusing knowledge between different stakeholders or platforms along the product lifecycle. In such applications, ontology (defined as an explicit specification of a conceptualization [32]) plays an important role in representing domain concepts. For example, Li et al. [33] propose a process knowledge graph for structuralizing a vast number of heterogeneous Computer-Aid Manufacturing (CAM) models (including both CAD models and NC processes) generated and stored in enterprises, so as to facilitate the sharing and reusing those models in NC process planning. An ontology is created to organize the inherent concepts/terms of heterogeneous CAM systems. Zhang et al. [34] proposed a graph-based knowledge reuse framework to facilitate decision making

in new product development. They create an ontology to construct concept maps which can capture and organize knowledge resources generated at various stages of product lifecycle, then develop an algorithm for knowledge navigation in new product development based on the concept maps. Another application of KGs in engineering design is anchored in that KGs can represent complex relations between entities which are very useful for design inspiration. For example, Sarica et al. [35] propose to use a technology KG, which covers sematic-level knowledge in all technology fields defined in the international patent classification system, to retrieve the existing engineering knowledge in a domain and discover engineering concepts around the domain for future design considerations and innovation. This approach has been used to facilitate cross-domain knowledge discovery in the design of flying cars [35]. Liang et al. [36] propose a knowledge graph framework based on the entity-relation model for representation of facts in indoor scene design. In their framework, spatial relations and dependencies between objects are organized in a knowledge graph to facilitate indoor scene design. The third typical application of KGs lies in big data management and systems integration. For example, Grangel-González [37] in his dissertation proposes a knowledge graph based approach for Industry 4.0. Entities and relations of Cyber-Physical-Systems (CPS) are identified and defined in his work. Hubauer et al. [38] present several use cases of industrial knowledge graph at Siemens, including data integration and federation across heterogeneous and distributed data sources, end-user oriented interaction components for semantic search, visualization, exploration of data, etc.

Besides the applications, the construction of KGs is another research thread in the KG literature where the focus is how a KG is built before it is used. Zhao et al. [39] describe that there are four key procedures in KG construction, namely, knowledge extraction, knowledge fusion, KG storage, and KG retrieval and visualization. Knowledge extraction is to extract knowledge from different data sources including structured data such as relational databases, semi-structured data such as HTML pages, and unstructured data such as free text and image etc. Knowledge fusion used to align the entities with the concepts in an ontology. KG storage used to store the KGs in computer-interpretable formats such as RDF or OWL files. KG retrieval and visualization used to search the entities/relationships using query languages such as SPARQL [40] and to visualize the KG using graphical tools. In the four procedures, knowledge extraction and knowledge fusion are the most challenging procedures since they are related to how the data is prepared and organized in a meaningful way. To tackle the challenges, Dong [41] reports their efforts at Amazon in constructing a product knowledge graph for dealing thousands of product verticals and a vast number of data sources. Their efforts are classified into four directions. The first direction is to harvest product knowledge from semi-structured sources on the web and from text product profiles using advanced extraction technologies such as the annotation-based extraction tool XPaths [42]. The second direction is to clean and integrate hands-off-the-wheel knowledge, by linking records that refer to the same real-world entity using machine learning techniques such as random forest models. The third direction is to conduct graph mining to decide the importance of entities and relationships, and

the fourth direction is applying knowledge learning with human in the loop. Other research contributions in KG construction are mainly focus on knowledge extraction using machine learning techniques and fixing the issues caused by automatic KG generation. For example, Liang et al. [36] develop a weakly supervised algorithm for extracting the knowledge graph representation from a small dataset using both structure and parameter learning. Zhao et al. [43] combine traditional natural language processing (NLP) techniques and deep learning methods to automatically extract triples from large unstructured Chinese text and construct an industrial knowledge graph in the automobile field. In order to address the issue of incompleteness (e.g., many triples are missing) occurs in many automatically constructed industrial KGs, Ringsquandl et al. [24] propose an event-based knowledge learning approach for KG completion.

In this paper, we propose a KG-based approach for structuralizing a large number of word-based cDSP formulations (word formulations), providing a way for designers (as decision makers) to efficiently navigate and reuse existing knowledge from the platform to support their decisions in design. It is noted that Zhang et al. [34] propose a KG-based approach for decision support in new product development. The limitation of their approach is anchored in that they only capture the product lifecycle data which limits its capability in supporting decisions. We overcome that limitation in our approach by capturing not only the lifecycle knowledge, but also the decision- and domain- related knowledge (see the meta-model in Section 3), which allow us to provide better decision support.

3. Method

In this section we present a method for building decision knowledge graph (DKG) and navigating decision knowledge. The method consists of three main blocks, namely, meta-model of DKG (mDKG) definition, DKG construction, and decision knowledge navigation (DKN) implementation, as shown in Figure 5. In this paper, a DKG is defined as a graph representation of a set of cDSPs whose key concepts, relationships are represented and recorded as triples at the computational level.

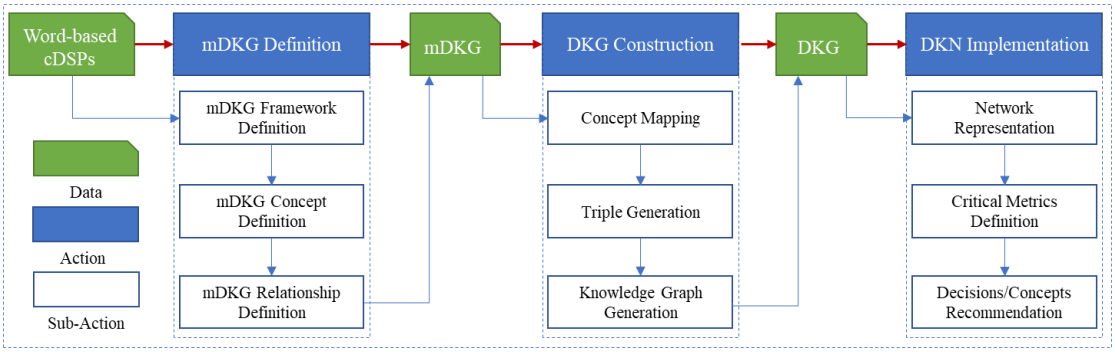

Figure 5 Overall process of the proposed method

The mDKG definition is a fundamental part of the proposed method which is used to define

the concepts and relationships involved into DKG. The key concepts and relationships of the mDKG are defined in Section 3.1. The DKG construction is the core step of the proposed method, and a pipeline of NLP process and steps for building DKG from word-based cDSPs are detailed in Section 3.2. The DKN implementation is the part that facilitate decision-makers finding useful decision knowledge, and it is implemented by navigating the constructed DKG, which are illustrated in Section 3.3.

3.1 mDKG Definition

The mDKG is important part of DKG since it determines what information and knowledge are represented in the DKG. In this section we first illustrate the structure of the mDKG, and then define the key concepts and relationships.

(1) The structure of mDKG

Since the goal of the DKG is to facilitate decision-makers finding useful knowledge, it must contain information about decision knowledge itself and therefore we define a decision dimension in the mDKG. Besides, the decision knowledge must have connections with domain-specific concepts as well as human (subjects) in that domain, which is helpful for locating decision knowledge. Therefore, the defined mDKG contains three dimensions, namely, decision, domain, and human. As shown in Figure 6, the yellow ellipses indicate the entities in a DKG, the rectangles indicate the concepts in the mDKG and the arrows indicate the relationship. In the mDKG, the decision dimension contains the fundamental concepts used to represent decision knowledge, the domain dimension contains the concepts in a specific domain, and the human dimension contains the concepts to describe the subjects related to te decision. All the concepts are illustrated in Table 1 and the relationships are illustrated in Table 2.

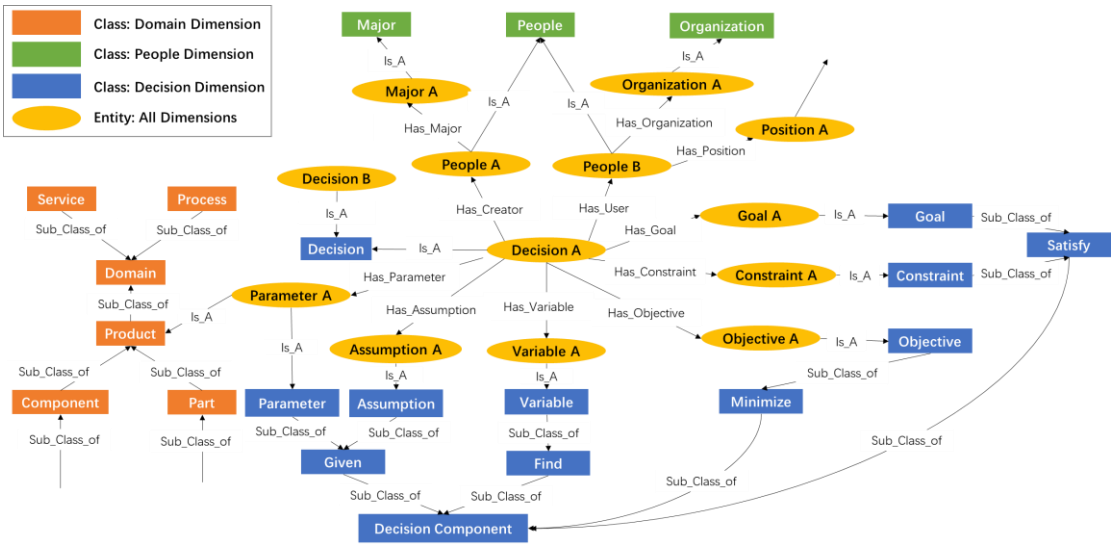

Figure 6 The main concepts and relationships of mDKG

(2) The concepts in mDKG

Ming and coauthors [21] provide an ontology for describing a cDSP. This ontology is adopted in this work to define the decision dimension (we eliminate some concepts that are not useful for navigating decision-related knowledge). There are four concepts contained in the decision dimension: *Given, Find, Satisfy,* and *Minimize,* which are the perspectives where decision-makers can seek knowledge. The *Parameter, Variable, Goal, Constraint, Assumption, and Objective* are used to further define these four perspectives.

The domain dimension is to organize concepts within a specific domain and provide a standard vocabulary for describing the decision dimension. For an industrial enterprise, the job of a decision-maker is typically to determine the parameters in design, manufacturing, and service, which imply the sources where they search for knowledge. Hence we define three concepts in the domain dimension of mDKG, namely, *Product*, *Process*, and *Service*. *Part* is used to further describe the *Product*. Similarly, *Step* and *Activity* are used to further describe *Process* and *Service respectively*. When building DKG for a specific domain, the domain dimension should be extended to include all concepts. For the human dimension, the mDKG contains three fundamental concepts, namely, *Position*, *Major*, and *Organization* to describe human-related properties.

Table 1 The concepts in mDKG

Dimensions	First Level	Second Level	Explanation of Concepts
Decision	*Given*	*Parameters*	*The value is fixed during the solution of the problem*
		Assumptions	*It's the base to construct the cDSP model*
	Find	*Variables*	*Its value changes constantly in the process of solving the problem*
	Satisfy	*Goals*	*Its dependent variable may violate a specified value or value direction*
		Constrains	*Its dependent variable cannot violate a specified value or value direction*
	Minimize	*Objective*	*Deviation of model results from targets under constrained conditions*
Domain	*Product*	*Part*	*It describes which part of the product the cDSP model relevant to*
	Process	*Step*	*It describes where the model is used in the product life cycle*
	Service	*Activity*	*It describes in which activities the product is used*
People	*Position*		*It describes what position the people relevant to the model have*
	Major		*It describes what major the people relevant to the model have*
	Organization		*It describes what organization the people relevant to the model belong to*

(3) The relationships in mDKG

The relationships between the concepts in Table 1 are also critical for mDKG. We define a list of relationships corresponding to the three dimensions in this section. As shown in Table 2, the name, domain, range of the relationships are listed.

Table 2 The concepts in mDKG

Name of Relationship	Domain	Range
has_given	*Decision*	*Given*
has_find	*Decision*	*Find*
has_satisfy	*Decision*	*Satisfy*
has_minimize	*Decision*	*Minimize*
has_assumption	*Decision, Given*	*Assumptions*
has_parameter	*Decision, Given*	*Parameters*
has_variable	*Decision, Find*	*Variables*
has_goal	*Decision, Satisfy*	*Goals*
has_constraint	*Decision, Satisfy*	*Constraints*
has_objective	*Decision, Minimize*	*Objective*
has_people	*Decision*	*People*
has_major	*People*	*Major*
has_organization	*People*	*Position*
has position	*People*	*Organization*
has_activity	*Service*	*Activity*
has_step	*Process*	*Step*
has_part	*Product*	*Part*
has_concept	*All*	*Domain*

3.2 DKG Construction

Based on the defined mDKG, a DKG can be constructed using advanced NLP methods. The input of this step is the mDKG and a number of word-based cDSPs in a specific domain, and the output is a constructed DKG, in which the knowledge in these word-based cDSPs is embodied. As shown in Figure , given a word-based cDSP, three steps are conducted to build a DKG, including concept mapping, triple generation and knowledge graph generation.

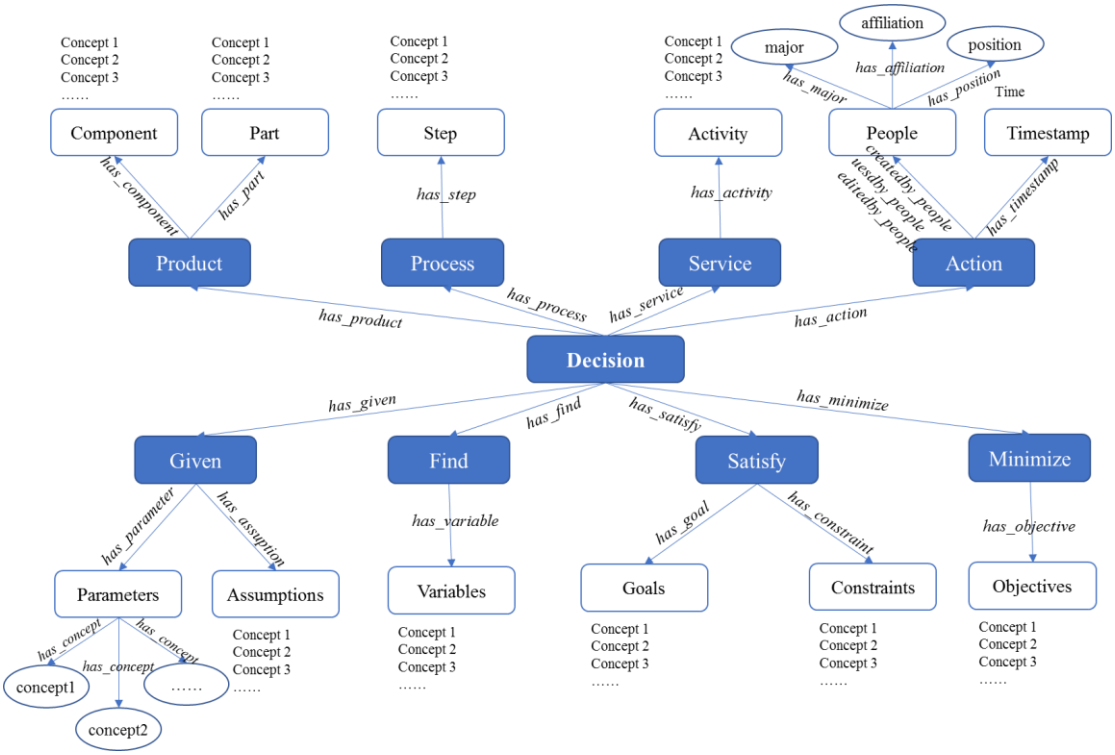

Figure 7 concept-based cDSP (ccDSP) tree structure

(1) Concept mapping

The input of this step is a list of word-based cDSPs, while the output is the corresponding concept-based cDSPs (ccDSP), which is represented by a tree structure presented in Figure . The ccDSP has two parts: the core part (the lower part in Figure) and the auxiliary part (the upper part in Figure), and it can be generated by a pipeline of NLP operations to the word-based cDSP, including initialization, word segmentation, block identification, importance evaluation, concept selection and ccDSP generation.

A. Initialization.

In this step a blank table is built, which is used to record all word-based cDSPs and the related concepts in the following steps. Each row of the table represents a cDSP and each column represents a concept. The value in each cell of the table measures the importance of the corresponding concept against the cDSP.

B. Word segmentation.

In this step all text in a word-based cDSP are segmented to get a list of words or phrases. This step is implemented based on the open source package – Natural Language Toolkit (NLTK[2]), and all the concepts defined in the mDKG are added to the dictionary of NLTK.

[2] https://github.com/nltk

C. Block identification.

The blocks *given, satisfy, find*, minimize and the sub-blocks *parameters, assumptions, variables, constraints, goals, minimize, objective* in the word-based cDSP are identified by a rule-based method. To implement that, we define a list of text rules using regular expression (RegEx), which is a sequence of characters that define a text search pattern.

D. Importance evaluation.

In this step the importance of a concept against a cDSP is evaluated and update the table created in Step A. It' s necessary because the importance of different concepts against a cDSP is different. For example, when a cDSP contains two concepts and one concept emerges in only 5 cDSPs of all 100 cDSPs while another emerges in 90 cDSPs of all 100 cDSPs, we can infer that the first concept is much more important than the second one for that cDSP. In this step, we use the Term Frequency – Inverse Document Frequency (TF-IDF) [44] as an importance indicator that is commonly used to measure the importance of a word against a document.

E. Concepts selection.

This step is to select important concepts based on their TF-IDF value. For the core part, the concepts under *Given, Find and Satisfy* are selected from each block based on their TF-IDF values. For the auxiliary part, the concepts under *Product, Process, Service,* and *People* are selected from all blocks. For example, to select concepts for *Product,* we find all concepts that are defined as the subclass of *Product* and then determine whether to put these concepts under *Product* according to their TF-IDF values. Therefore, the key is to set up a threshold value of TF-IDF for selecting concepts. In this work, it is impossible to set up a fixed threshold value for all cDSPs. An extreme example is that when there is only one concept under a block, we still put this concept under the block although the TF-IDF value is low. We develop a strategy for concepts selection, which follows the following two rules:

- When the number of concepts in a block is small (< 5), it is impossible to establish a probability distribution model, so a probability distribution model is established with the whole concept of cDSP. 5 is statistically the threshold of whether the quantity of a set of data can constitute an effective distribution. We simply select concepts with TF-IDF values higher than a threshold θ_1. The threshold θ_1 is calculated through the following steps.
  - Assuming all the TF-IDF values of the concepts in cDSP follows an exponential distribution with parameter λ_1, we define the threshold value is $\theta_1 = {}^{1}/_{\lambda_1} + {}^{1}/_{\lambda_1{}^{2}}$.
  - When there is no concept with TF-IDF that is larger than the threshold we update the threshold to $\theta_1 = {}^{1}/_{\lambda_1}$.

- When the number of concepts in a block is large enough ($\geq$ N), we can establish a probability distribution model with these concepts. We simply select concept with TF-IDF values higher than a threshold θ_2. The threshold θ_2 is calculated through the following steps.
  - Assuming all the TF-IDF values of a block follow an exponential distribution with parameter λ_2, and the mean and variance are $1/\lambda_2$ and $1/\lambda_2{}^2$ respectively. We define the threshold value is $\theta_2 = 1/\lambda_2 + 1/\lambda_2{}^2$.
  - When there is only very small number (normally $\leq$2) of concepts have TF-IDF that are higher than the threshold, we update the threshold to $\theta_2 = 1/\lambda_2$.

E. Concept-based cDSP generation.

This step generates the concept-based cDSP for all word-based cDSPs using the selected concepts in Step E.

(2) Triple generation

Due to the fact that knowledge graph is a group of semantically connected concepts, the most important work to build a knowledge graph is to generate a list of triples. A triple describes the relationship between a pair of concepts and is represented by the form *Triple<Entity, Relationship, Entity>*. Here, the Entity and Relationship are defined in mDKG. Based on the generated ccDSPs constructed in the previous step, this work adopts Depth First Search (DFS) as the fundamental algorithm to traverse the whole ccDSP to generate a list of triples. To clearly explain the procedure, we provide the pseudo code based on a recursive mechanism.

Pseudo Code 1: Triple Generation

```
Triples = TripleGeneration (ccDSP, Point)

% Input: ccDSP: this is a concept-based cDSP

% Input: Point: this is a starting node for triple generation

% Output: Triples: a list that contains all triples

SearchedPoint = [] % this is an array to record the searched points or vertices in ccDSP

Triples = [] % this is an array to record all found triples

AdjacentPoints = findAdjacentPoints(ccDSP, Point)

For each adjacentPoint in AdjacentPoints Do

    If adjacentPoint is not in SearchedPoint Then

        tempTriple = TripleGeneration (ccDSP, adjacentPoint)
```

```
            Triples = [Triples, tempTriple]

        End

End
```

(3) Knowledge graph generation

In this work, we adopt Neo4J which is a commonly used graph database to create and store the DKG. Neo4J provides a high-level query language enabling us to operate the knowledge graph stored in Neo4J. We build the knowledge graph by two steps, including node generation and edge generation.

A. Node generation.

A node is to represent a concept in the DKG and an entity in a triple. We first build a list of entities from all triples by removing duplicate entities. Then, only a single node is created for each entity in the list by the following pseudo code.

Pseudo Code 2 : Node Generation

```
CreateNodes ( mDKG, EntityList)

% Input: EntityList : the list that contain all concepts that used in the triples

% Input: mDKG : the meta-model of DKG

EntityID = 1 % Initialize the id for node in DKG

ExistEntity = [] % record all nodes that already in DKG

For each Entity in EntityList Do

    If Entity is not in ExistEntity Then

        EntityType = FindEntityType(mDKG, Entity) % find the class of the entity by mDKG

        QueryCode = (EntityID:EntityType[name:Entity]) % generate Cypher query sentence

        Execute(QueryCode) % Execute the query sentence

        EntityID = EntityID + 1

    End

End
```

B. Edge generation.

An edge is to represent a relationship in the DKG. We create all edges by iteratively generate

edge generation code *(EntityID--Relationship-->EntityID)* and execute the code. After that a DKG can be generated, and in this DKG all decision knowledge is connected together for further exploration.

3.3 DKN Implementation

With the constructed DKG, several knowledge service can be implemented, such as semantic retrieval, knowledge recommendation, and decision question answering. In this section, a method for navigation decision knowledge is developed, the goal of DKN is to locate intended decision as soon as possible. The DKN is regarded as a divergence and convergence process. The divergence process is to recommend a batch of related concept for decision-makers to further filter the decision, while the convergence process is to recommend decisions based on the semantic relationships. The basic process of DKN is shown in Figure 8.

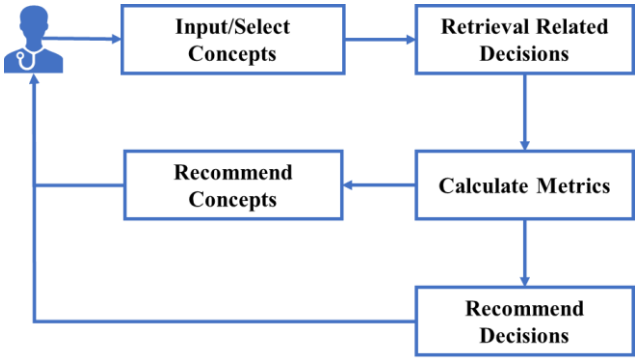

Figure 8 The basic procedures of DKN

As shown in Figure 8, after a decision-maker inputting the key concept and condition, 1) the related decisions and domain concepts are retrieved, then 2) a group of metrics are defined to measure these decisions and concepts for recommendation, and finally 3) the related decision and concepts are provided to decision-makers.

We use the concepts and conditions given by decision-makers to retrieve related decisions and its related concepts based on Neo4J query language. The query result is transferred into a network in which the nodes represent all concepts and the edge represent the weights of the relations, as shown in **Error! Reference source not found.**. Here the weight means the importance of a concept against a decision, which is measured by the TF-IDF value calculated in Section 3.2.

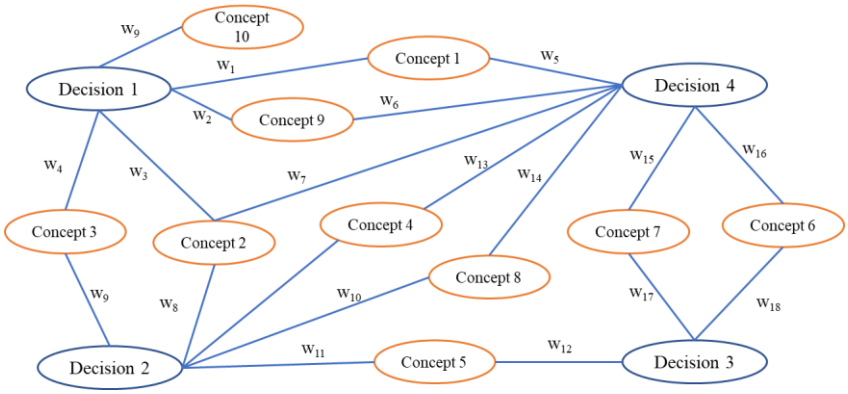

Figure 9 An example network of the retrieval results containing 4 decisions and 9 concepts

With this network, three metrics are defined to rank decisions and concepts for recommendation, including degree centrality (DC), number of common concepts (NCC) and sum distance (SD).

(1) The DC indicates the number of decisions that a node directly connects to. For example, "Decision 1" has a DC of 2 since it connects directly to "Decision 4" and "Decision 2", and "Concept 2" has a DC of 3 since it connects directly to "Decision 1", "Decision 2" and "Decision 4". Here, the direct connection means two decisions are linked through the pattern "Decision - Concept - Decision".

(2) The NCC indicates the number of common concepts that a decision directly connects to, and the common concepts are the concepts connect at least two different decisions. For example, "Decision 1" has an NCC of 4 since it connects directly to "Concept 1", "Concept 2", "Concept 3", and "Concept 9". Here "Concept 10" is not a common concept since it only connects to "Decision 1".

(3) The SD indicates the closeness of a decision with its connecting decisions, and this metric is calculated by summing the weights of all relations that connecting the decisions. For example, the SD of "Decision 1" is first sum all the weights between "Decision 1" and the directly connected concepts (w_1+w_2+w_3+w_4), and then sum all the weights between these concepts and their directly connected decisions (w_5+w_6+w_7+w_8+w_9). For a decision with M connected concepts, which are connected with N decisions, the SD can be calculated by the following Equation 1.

$$\mathrm{SD} = \sum_{j=1}^{M} w_j + \sum_{j=1}^{M} \sum_{k=1}^{N} w_{jk} \quad (1)$$

With these defined metrics, the osculation value method is adopted to convert multiple metrics into one comprehensive metric C_i, and then all nodes are ranked based on C_i. The osculation value method calculates the comprehensive metric through the following steps.

A. Build metrics matrix

This step is to build a matrix to contain all values of multiple metrics, as shown in Equation 2:

$$a = \begin{bmatrix} a_{11} & a_{12} \dots & a_{1m} \\ \dots & \dots & \dots \\ a_{21} & a_{22\dots} & a_{2m} \\ \dots & \dots & \dots \\ a_{n1} & a_{n2} \dots & a_{nm} \end{bmatrix} \quad (2)$$

where n is the number of decisions and concepts, m=3 is the number of metrics, a_{ij} is the value of metrics.

B) Uniform metrics

The metrics in matrix *a* are divided into two types: positive metrics and negatives metrics. Positive metrics mean the bigger they are, the better the result is. For example, a bigger DC means a decision or a concept has more connection with others. Negative metrics mean the smaller they are, the better the result is. For example, a smaller SC means a decision or concept has a bigger distance with others.

In this step, we add '-' to negative metrics to achieve union. SC is uniformed in this step, as shown in Equation 3.

$$b = \begin{bmatrix} a_{11} & a_{12} \dots & -a_{1m} \\ \dots & \dots & \dots \\ a_{21} & a_{22\dots} & -a_{2m} \\ \dots & \dots & \dots \\ a_{n1} & a_{n2} \dots & -a_{nm} \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} \dots & b_{1m} \\ \dots & \dots & \dots \\ b_{21} & b_{22\dots} & b_{2m} \\ \dots & \dots & \dots \\ b_{n1} & b_{n2} \dots & b_{nm} \end{bmatrix} \quad (3)$$

C) Normalize matrix

Because different metrics have different scales, and the osculation value method transform metrics from many to one, so it's necessary to unify the scales. We achieve that by normalizing the *b* matrix. As shown in Equation 4.

$$r = \begin{bmatrix} r_{11} & r_{12} \dots & r_{1m} \\ \dots & \dots & \dots \\ r_{21} & r_{22\dots} & r_{2m} \\ \dots & \dots & \dots \\ r_{n1} & r_{n2} \dots & r_{nm} \end{bmatrix} (4)$$

$$r_{ij} = \frac{b_{ij}}{\left(\sum_{k=1}^{n} {b_{kj}}^2\right)^{\frac{1}{2}}}$$

$$r_{ij} = \frac{b_{ij}}{\left(\sum_{k=1}^{n} {b_{kj}}^2\right)^{\frac{1}{2}}}$$

$$\text{The best point } r_j^+ = \max(r_{ij})$$

$$\text{The worst point } r_j^- = \min(r_{ij})$$

$$\text{i} = 1,2,3,\dots,\text{n}$$

$$\text{j} = 1,2,3,\dots,\text{m}$$

D) calculate C_i

$$d_i^+ = \left[\sum_{j=1}^{m}\left(r_{ij} - r_j^+\right)^2\right]^{\frac{1}{2}}$$

$$d_i^- = \left[\sum_{j=1}^{m}\left(r_{ij} - r_j^-\right)^2\right]^{\frac{1}{2}}$$

i=1,2,3,...,n

j=1,2,3,...,m

Osculation Value

$$C_i = \frac{d_i^+}{d^+} - \frac{d_i^-}{d^-}$$

$$d^+ = \min(d_i^+),\ d^- = \max(d_i^-)$$

i=1,2,3,...,n

Smaller C_i is, the better the decision or concept is.

With C_i defined and calculated, the concepts and decision recommendation can be achieved in a computational manner.

4. Case study

We selected supply chain(SC) as the domain to demonstrated that our method is effective. Supply Chain is a network that extends from the supplier to the end consumer and as product passes

through network each stakeholder add some value and each facility forms a node in the supply network. Hence, there is a need to enable different stakeholders in the network to share and reuse the knowledge and provide better informed decisions.

In this section, 25 word-based cDSPs in the domain of supply chain management (SCM) are used to build a DKG. The data comes from papers online, and we transformed the model into the cDSP's form. In Table 3 we list the title of these cDSPs and the detailed documents can be downloaded from Github[3].

The first part describes what contents are in the knowledge graph. The second part shows a case which shows our method's method to find the decision. The third part is the proof of that our method have a better performance than traditional method. The final part is the analysis of why our method is better and the limitation of our method.

4.1 The process of building DKG

According to Section 3.2, we constructed a DKG of supply chain management (SCM) by implementing the process step by step. We then import all edges and nodes into Neo4J. Figure 10 shows a part of the constructed DKG, in which, the green nodes represent decisions, the pink nodes represent <*given, find, satisfy, minimize, people, timestamp, product, process, service*>, the yellow nodes represent <*parameters, assumptions, variables, goals, constraints, objectives, component, part, step, activity, major, organization, position*>, and the blue nodes represent the concepts. The whole figure can be download from Github[3], which contains 936 entities and 1458 relations totally. In Table 3, the number of decisions and concepts that connected with every decision are listed.

[3] https://github.com/zhaoleilei-19950226/Knowledge-Graph

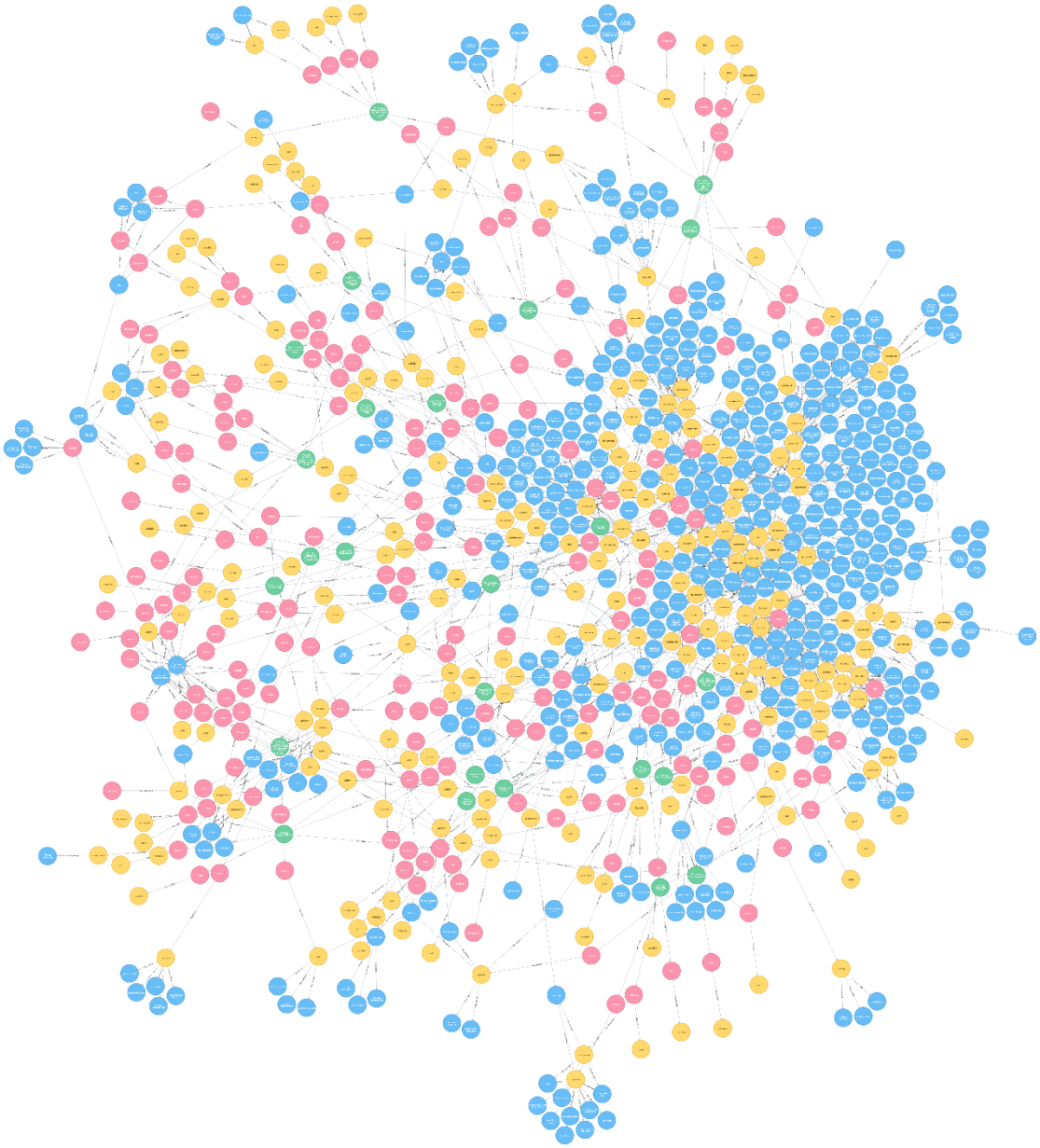

Figure 10 The constructed DKG based on 25 word-based cDSP in SCM domain

Table 3 The number of the concepts and decisions a model connected with

Number	Model's name	Decision	Concepts
Decision1	Arc-Based Multi-Echelon Supply Chain Model	23	26
Decision2	Big Screen TV Company Example	24	46
Decision3	Canon Closed Loop Supply Chain Network	24	35
Decision4	Customer Requirement Fulfillment Model	24	18
Decision5	Defects and Pricing Model	23	16
Decision6	Fuzzy Logistic Model	24	28
Decision7	Fuzzy Mathematical Model	24	34
Decision8	Geoffrion and Graves Distribution Model	23	13
Decision9	Integrated Customer Related Life Cycle Model	23	36
Decision10	Modelling a closed-loop supply chain with a heterogeneous fleet under carbon emission reduction policy	23	63
Decision11	Non-resilient and competitive SCND	24	30
Decision12	Optimal supply chain resilience with consideration of failure propagation and repair logistics	17	48
Decision13	Product Life Cycle Model	24	38
Decision14	Returning Vehicle Model	24	42
Decision15	Selecting a product family and designing its supply chain	20	21
Decision16	Similarity Pattern Model	22	14
Decision17	System Dynamics Model	23	41
Decision18	Tactical BOM Supply Chain Model	24	32
Decision19	Time-Windows Model	24	24
Decision20	Vehicle Routing Model	23	32
Decision21	Waste Collection Model	24	10
Decision22	Waste Pricing Model	24	30
Decision23	Web-Based Model	23	27
Decision24	Genetic Optimization Model	24	26
Decision25	The LP Model to Optimize the Biofuel Supply Chain	22	18

4.2 Feasibility of the navigation process

According to Section 3.3, we illustrate the navigation process step by step.

We tested all 25 decisions with our navigation system with following steps:

Assumption: Decision d has N concepts connected with;

1) Take each concept as the first input to find decision *d*;

2) We used top N method to define whether we found the decision. When we first input *ith* concept and with our navigation system decision *d* is ranked top 3, we regarded that we find the decision. And users have S_i steps interaction with the system.

3) We finally get the average steps $AS = \frac{\sum_{i=1}^{N} S_i}{N}$ to find decision *d*.

The Table 4 is the final result:

Table 4 The table of each decision's AS

Decision	AS	Decision	AS	Decision	AS	Decision	AS	Decision	AS
Decision1	1.58	Decision6	1.78	Decision11	1.39	Decision16	1.18	Decision21	1.91
Decision2	1.28	Decision7	1.40	Decision12	1.12	Decision17	1.30	Decision22	1.19
Decision3	1.27	Decision8	1.44	Decision13	1.28	Decision18	1.19	Decision23	1.00
Decision4	1.74	Decision9	1.77	Decision14	1.44	Decision19	1.55	Decision24	1.92
Decision5	1.50	Decision10	1.09	Decision15	1.23	Decision20	1.39	Decision25	1.42

We get an example as following to show our method to find decisions.

Taking Decision *Fuzzy Mathematical Model* as our target to find, and we select a very general concept "cost" as the starting point of the navigation process. We define that when the target decision is on the top 3 of the navigation list, the target decision is successfully found.

In each step we got a list of decisions and concepts with their osculation values. To proceed the navigation process, the concept with the smallest osculation value is selected as the next concept for the navigation. The navigation process is listed in Table 5-8.

First step input: cost

Table 5 The output of the first step' input

Decision	OC	Concept	OC
Decision23	0	cost	0
Decision22	0.37	transportation	34.19
Decision13	0.52	System Engineering	45.97
Decision20	0.62	delivery	63.88
Decision18	0.62	depots	64.19
Decision19	0.96	negative	69.13
Decision10	0.97	large number	69.59
Decision15	1.10	flow conservation	69.64

Decision11	1.11	customer demands	69.94
Decision21	1.36	distance	73.90
Decision9	1.49	stored product	74.05
Decision2	1.58	tardiness	74.13
Decision8	1.66	Fixed cost	74.13
Decision5	1.88	inventory	74.16
Decision3	1.88	Capacity restriction	74.31
Decision4	1.89	penalty	74.37
Decision24	1.93	remaining vehicles	74.49
Decision14	1.94	binary	74.53
Decision6	2.92	inventory	74.16
Decision7	3.11	Capacity restriction	74.31

Second step input: cost-transportation

Table 6 The output of second step's input

Decision	**OC**	**Concept**	**OC**
Decision23	0	cost	0.11
Decision10	0.40	transportation	3.13
Decision19	0.65	delivery	67.80
Decision18	0.80	Hamed Fazlollahtabar	83.27
Decision3	0.95	System Engineering	87.33
Decision14	0.96	distance	90.68
Decision24	0.98	penalty	91.10
Decision11	0.99	flow conservation	91.95
Decision21	1.51	storage	96.89
Decision8	1.81	stored product	98.55
Decision2	1.94	depots	98.81
Decision7	3.30	Capacity restriction	98.94
Decision6	3.88	customer demands	98.95

Third step input: cost-transportation-delivery

Table 7 The output of the third step's input

Decision	OC	Concept	OC
Decision14	0	transportation	-0.56
Decision23	1.045	cost	-0.45
Decision10	1.70	delivery	-0.43
Decision2	3.44	distance	249.33
Decision7	5.34	stored product	311.97
Decision6	10.49	flow conservation	312.10

Forth steps input: cost-transportation-delivery-distance

Table 8 The output of the forth step's input

Decision	OC	Concept	OC
Decision23	0	distance	-0.61
Decision10	0.46	delivery	-0.61
Decision7	1.04	cost	-0.61
Decision6	4.24	transportation	-0.57

In this case, we finally found the decision *Fuzzy Mathematical Model* through 4 steps.

4.3 Comparison with conventional methods

In this section, the effectiveness of the navigation process is verified by comparing the proposed method with traditional keywords retrieval method (traditional method). The essential difference of these two methods is that the proposed method uses the semantical structure that captured by the DKG to find target decisions, while the traditional keywords retrieval method only use the matching of characters to find target decisions. An indicator called *hit ratio (HR)* is defined to evaluate these two methods. For the experiment, assuming the user know *n* concepts for finding a specific decision *d*, and he/she select m concepts as combination to retrieve the decision, we define that the target decision is found when the it is in the top *k* of the retrieval list. By this way, the HR is defined as following equation.

$$HR = \frac{s}{A_n^m}$$

We adapted HR_f as the *HR* of the proposed method and HR_s as the *HR* of the traditional method. Totally, 18 experiments under different configurations of <*n, m, k*> are conducted to evaluate the method. In each experiment we get the corresponding HR_f and HR_s. If $HR_f > HR_s$ obviously, it proves our method is efficient.

We adapted hypothesis testing to test whether our method is more efficient.

1) First we construct a variable $\Delta\mathrm{HR}$ to evaluate whether our method is better. The variable is defined as follow:

$$\Delta\mathrm{HR} = \frac{HR_f - HR_s}{HR_s}$$

This variable measure how much our method improve the HR compared with the traditional method. If $\Delta\mathrm{HR} > 0$, it proves our manner is better.

2) Then we can get the null hypothesis $\Delta\mathrm{HR} \leq 0$ and the alternative hypothesis $\Delta\mathrm{HR} > 0$.

3) Next we construct test statistic *t*.

$$\mathrm{t} = \frac{\Delta\mathrm{HR}}{s/\sqrt{c-1}}$$

s is $\Delta\mathrm{HR}'\mathrm{s}$ standard deviation and *c* is the number of samples.

If $\mathrm{t} \geq t_\alpha(c-1)$, we can reject the null hypothesis and accept the alternative hypothesis with a confidence 1-α.

We got the $\Delta\mathrm{HR} \leq 0$ of each decision and did the hypothesis testing. The following tables 9&10 are the results of t values and confidence correspond to them. The original data is in the attachment.

Table 8 The result of the t values

k	n	6			7		
	m	2	3	4	2	3	4
1		1.62	1.31	1.43	1.44	1.44	1.58
2		1.78	1.31	1.43	1.44	1.44	1.58
3		1.25	1.14	0.95	1.44	1.44	1.58

Table 9 The confidence corresponds to the t values

k	n	6			7		
	m	2	3	4	2	3	4
1		90%	85%	90%	90%	90%	90%
2		95%	85%	90%	90%	90%	90%
3		85%	85%	80%	90%	90%	90%

All most the t value is bigger than $t_{0.15}(24) = 1.059$ or $t_{0.1}(24) = 1.318$. It indicates that our method can find the corresponding decision more effectively with a confidence of 85% or 90% than without our method.

4.4 Discussion

In this section we test the feasibility of our method for constructing a knowledge graph using 25 word-based cDSPs, and the results of which shows that the method support converting textual documents to structured knowledge graphs. In order to test the performance of the knowledge graph in the navigation of wanted knowledge, we select a decision as a target in the 25 cDSPs and see if the designer can quickly find it. Results show that the wanted knowledge is found after 4 steps. Finally, we compare the knowledge-graph-based method with traditional keyword-based searching method and observe that the former has better performance in terms of both effectiveness and efficiency in navigating the wanted knowledge. One limitation of our knowledge-graph-based method is anchored in that the navigation process rely on a complete knowledge graph as a support. If the knowledge-graph is not complete, the performance of navigation will be affected.

This knowledge graph can be easily expanded to other domains such as products and manufacturing systems design due to the generality of the mDKG model. Rapid knowledge navigation can reduce the time decision makers spend on searching information, which as a result can accelerate design processes.

5. Conclusion

Enabling designers to quickly and precisely searching relevant information is critical for supporting their decisions during design processes, especially in an environment where designers are confronted with huge amount of unstructured textual documents. In this paper we propose ~~to use~~ knowledge graph as a means to represent word-based cDSP formulations (where decision-related knowledge is formulated in a linguistic manner) and facilitate rapid navigation of the target knowledge for supporting decisions. Our contributions in this paper are three-fold. First, we develop a meta-model for representing the cDSPs as a decision knowledge graph (mDKG), and formally define the concepts and relationships of the mDKG. The mDKG provides a common knowledge template for designers, which makes the decision-related knowledge shared and reused more easily. Second, we propose a method for automatically converting word-based cDSPs to knowledge graph through natural language processing. cDSPs are transformed into knowledge graph by word segmentation, regularization etc. and semantic information extraction of concept by tf-idf is incorporated into the knowledge graph. This is critically important because it saves a lot of time spent on manually constructing knowledge graphs which is typically necessary for structuring the textual data. Third, we develop a divergence-convergence-based procedure for rapidly and accurately navigating decision-related knowledge. Osculation value method associated with tf-idf value is adopted in this research which makes the system can better understand the designer's intentions. The mDKG method is tested through a scenario where a designer is trying the navigate the target knowledge within 25 word-based cDSPs in the supply chain design domain. The

comparison of our method with traditional keyword-based searching method indicates that out method has better performance in terms of both effectiveness and efficiency.

Our future work is targeted at expanding the current knowledge graph to other domains including products design, manufacturing systems design, etc., and deploy the knowledge graph to the CBDS platform so that it can provide decision support to designers of different domains. Using machine learning algorithms to enable adaptive questing answering based on the knowledge graph is an additional future research opportunity.

Acknowledgement

Jia Hao acknowledges the support from National Ministries Foundation (JCKY20192019204B021 and 2019-JCJQ-ZD-049-02). Zhenjun Ming acknowledges the support from National Natural Science Foundation of China (51805033) and Beijing Institute of Technology Research Fund Program for Young Scholars (3030011182037).

References

[1] Jing, L., Jiang, S., Li, J., Peng, X., and Ma, J., 2020, "A cooperative game theory based user-centered medical device design decision approach under uncertainty," Adv Eng Inform, 47(2021), p. 101204.

[2] Ming, Z., Sharma, G., Allen, J. K., and Mistree, F., 2019, "Template-based configuration and execution of decision workflows in design of complex engineered systems," Adv Eng Inform, 42, p. 100985.

[3] Shahtaheri, Y., Flint, M. M., and Garza, J. M. d. l., "A multi-objective reliability-based decision support system for incorporating decision maker utilities in the design of infrastructure," Adv Eng Inform, 42, pp. 100939-100939.

[4] Mistree, F., Smith, W. F., Bras, B. A., Allen, J. K., and Muster, D., 1990, "Decision-Based Design: a Contemporary Paradigm for Ship Design," Transactions of the Society of Naval Architects and Marine Engineers, 98, pp. 565-597.

[5] Li, Z., Raskin, V., and Ramani, K., 2008, "Developing Engineering Ontology for Information Retrieval," J Comput Inf Sci Eng, 8(1), p. 011003.

[6] Mcmahon, C., Lowe, A., Culley, S., Corderoy, M., and Stewart, D., 2004, "Waypoint: An Integrated Search and Retrieval System for Engineering Documents," J Comput Inf Sci Eng, 4(4), pp. 264-264.

[7] Kroetsch, M., and Weikum, G., 2016, "Journal of Web Semantics: Special Issue on Knowledge Graphs," http://www.websemanticsjournal.org/index.php/ps/announcement/view/19.

[8] Dolšak, B., and Novak, M., 2011, "Intelligent decision support for structural design analysis," Adv Eng Inform, 25(2), pp. 330-340.

[9] Jiang, H., Kwong, C. K., Okudan Kremer, G. E., and Park, W. Y., 2019, "Dynamic modelling of customer preferences for product design using DENFIS and opinion mining," Adv Eng Inform, 42, p. 100969.

[10] Farhang Mehr, A., and Tumer, I. Y., 2006, "Risk-Based Decision-Making for Managing Resources During the Design of Complex Space Exploration Systems," J Mech Design, 128(4), pp. 1014-1022.

[11] Arendt, J. L., McAdams, D. A., and Malak, R. J., 2012, "Uncertain Technology Evolution and Decision Making in Design," J Mech Design, 134(10).

[12] Mistree, F., Marinopoulos, S., D., J., and Shupe, J. A., 1988, "The Design ol Aircraft using the Decision Support Problem Technique," NASA Conrracror Report 4134.

[13] Bascaran, E., Bannerot, R. B., and Mistree, F., 1989, "Hierarchical Selection Decision Support Problems in Conceptual Design," *Engineering Optimization*, 14(3), pp. 207-238.

[14] Mistree, F., Hughes, O. F., and Bras, B., 1993, "Compromise Decision Support Problem and the Adaptive Linear Programming Algorithm," Progress in Astronautics and Aeronautics, 150, pp. 251-251.

[15] Smith, W. F., Kamal, S., and Mistree, F., 1987, "The Influence of Hierarchical Decisions on Ship Design," Marine Technology, 24(2), pp. 131-142.

[16] Mistree, F., and Langley Research, C., 1988, The Design of aircraft using the decision support problem technique, Washington, D.C.: National Aeronautics and Space Administration, Scientific and Technical Information Division.

[17] Seepersad, C. C., Allen, J. K., McDowell, D. L., and Mistree, F., 2006, "Robust design of cellular materials with topological and dimensional imperfections," J Mech Design, 128(6), pp. 1285-1297.

[18] Bascaran, E., Mistree, F., and Bannerot, R. B., 1987, "Compromise: An effective approach for solving multiobjective thermal design problems," *Engineering Optimization*, 12(3), pp. 175-189.

[19] Nellippallil, A. B., Rangaraj, V., Gautham, B. P., Singh, A. K., Allen, J. K., and Mistree, F., 2018, "An Inverse, Decision-Based Design Method for Integrated Design Exploration of Materials, Products, and Manufacturing Processes," J Mech Design, 140(11), pp. 111403-111403-111417.

[20] Panchal, J. H., Fernández, M. G., Paredis, C. J. J., and Mistree, F., 2004, "Reusable Design Processes via Modular, Executable, Decision-Centric Templates," AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference.Albany, NY. Paper Number AIAA-2004-4601.

[21] Ming, Z., Yan, Y., Wang, G., Panchal, J. H., Goh, C.-H., Allen, J. K., and Mistree, F., 2016, "Ontology-based executable design decision template representation and reuse," AI EDAM - Artificial Intelligence for Engineering Design Analysis and Manufacturing, 30(4), pp. 390-405.

[22] Ehrlinger, L., and Wöß, W., 2016, "Towards a Definition of Knowledge Graphs," SEMANTiCS (Posters, Demos, SuCCESS), 48.

[23] Paulheim, H., 2017, "Knowledge graph refinement: A survey of approaches and evaluation methods," Semantic web, 8(3), pp. 489-508.

[24] Ringsquandl, M., Kharlamov, E., Stepanova, D., Hildebrandt, M., Lamparter, S., Lepratti, R., Horrocks, I., and Kröger, P., "Filling gaps in industrial knowledge graphs via event-enhanced embedding," Proc. 17th International Semantic Web Conference, CEUR-WS. org.

[25] Wang, P., Jiang, H., Xu, J., and Zhang, Q., 2019, "Knowledge graph construction and applications for Web search and beyond," Data Intelligence, 1(4), pp. 333-349.

[26] Wang, X., He, X., Cao, Y., Liu, M., and Chua, T.-S., "Kgat: Knowledge graph attention network for recommendation," Proc. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 950-958.

[27] He, L., Shao, B., Xiao, Y., Li, Y., Liu, T.-Y., Chen, E., and Xia, H., 2018, "Neurally-guided semantic navigation in knowledge graph," IEEE Transactions on Big Data.

[28] Bordes, A., Chopra, S., and Weston, J., 2014, "Question answering with subgraph embeddings," arXiv preprint arXiv:1406.3676.

[29] Ringsquandl, M., Lamparter, S., Brandt, S., Hubauer, T., and Lepratti, R., "Semantic-guided feature selection for industrial automation systems," Proc. International Semantic Web Conference, Springer, pp. 225-240.

[30] Kharlamov, E., Mailis, T., Mehdi, G., Neuenstadt, C., Özçep, Ö., Roshchin, M., Solomakhina, N., Soylu, A., Svingos, C., and Brandt, S., 2017, "Semantic access to streaming and static data at Siemens," Journal of Web Semantics, 44, pp. 54-74.

[31] Hachey, B., Radford, W., Nothman, J., Honnibal, M., and Curran, J. R., 2013, "Evaluating entity linking with wikipedia," Artif Intell, 194, pp. 130-150.

[32] Gruber, T. R., 1993, "A Translation Approach to Portable Ontology Specifications," Knowledge Acquisition, 5(2), pp. 199-220.

[33] Li, X., Zhang, S., Huang, R., Huang, B., Xu, C., and Kuang, B., 2018, "Structured modeling of heterogeneous CAM model based on process knowledge graph," The International Journal of Advanced Manufacturing Technology, 96(9-12), pp. 4173-4193.

[34] Zhang, C., Zhou, G., Lu, Q., and Chang, F., 2017, "Graph-based knowledge reuse for supporting

knowledge-driven decision-making in new product development," Int J Prod Res, 55(23), pp. 7187-7203.

[35] Sarica, S., Song, B., Luo, J., and Wood, K., "Technology Knowledge Graph for Design Exploration: Application to Designing the Future of Flying Cars," Proc. ASME 2019 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers Digital Collection.

[36] Liang, Y., Xu, F., Zhang, S.-H., Lai, Y.-K., and Mu, T., 2018, "Knowledge graph construction with structure and parameter learning for indoor scene design," Computational Visual Media, 4(2), pp. 123-137.

[37] Grangel-González, I., 2019, "A Knowledge Graph Based Integration Approach for Industry 4.0," Universitäts-und Landesbibliothek Bonn.

[38] Hubauer, T., Lamparter, S., Haase, P., and Herzig, D. M., "Use Cases of the Industrial Knowledge Graph at Siemens," Proc. International Semantic Web Conference (P&D/Industry/BlueSky).

[39] Zhao, Z., Han, S.-K., and So, I.-M., 2018, "Architecture of Knowledge Graph Construction Techniques," International Journal of Pure and Applied Mathematics, 118(19), pp. 1869-1883.

[40] Pérez, J., Arenas, M., and Gutierrez, C., 2009, "Semantics and complexity of SPARQL," ACM Transactions on Database Systems (TODS), 34(3), pp. 1-45.

[41] Dong, X. L., "Challenges and innovations in building a product knowledge graph," Proc. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2869-2869.

[42] Lockard, C., Dong, X. L., Einolghozati, A., and Shiralkar, P., 2018, "CERES: distantly supervised relation extraction from the semi-structured web," Proceedings of the VLDB Endowment, 11(10), pp. 1084-1096.

[43] Zhao, M., Wang, H., Guo, J., Liu, D., Xie, C., Liu, Q., and Cheng, Z., 2019, "Construction of an industrial knowledge graph for unstructured Chinese text learning," Applied Sciences, 9(13), p. 2720.

[44] Hao, J., Yan, Y., Gong, L., Wang, G., and Lin, J., 2014, "Knowledge map-based method for domain knowledge browsing," Decision Support Systems, 61, pp. 106-114.

[45]https://medium.com/comet-ml/using-fasttext-and-comet-ml-to-classify-relationships-in knowledge-graphs-e73d27b40d67