QUESTION GENERATION FROM KNOWLEDGE GRAPH WITH PRESCRIBED DIFFICULTY LEVEL

Warda Rizwan

Enrollment No: 01-241171-041

Supervisor: Dr Raja Muhammad Suleman

A thesis submitted to the Department of Software Engineering, Faculty of Engineering Sciences, Bahria University, Islamabad in the partial fulfillment for the requirements of a Masters degree in Software Engineering

October 2019

Approval Sheet
Thesis Completion Certificate

Scholar's Name: **Warda Rizwan** Registration No: **01-241171-041**

Program of Study: **MS Software Engineering**

Thesis Title: **Question Generation from Knowledge Graph with Prescribed Difficulty Level**

It is to certify that the above student's thesis has been completed to my satisfaction and, to my belief, its standard is appropriate for submission for Evaluation. I have also conducted plagiarism test of this thesis using HEC prescribed software and found similarity index at _____that is within the permissible limit set by the HEC for the MS/MPhil degree thesis. I have also found the thesis in a format recognized by the BU for the MS/MPhil thesis.

Principal Supervisor's Signature: _____

Date: **19 October, 2019** Name: **Dr. Raja M. Suleman**

Certificate of Originality

This is certified that the intellectual contents of the thesis **Question Generation from Knowledge Graph with Prescribed Difficulty Level** are the product of my own research work except, as cited property and accurately in the acknowledgements and references. The material is taken from sources as research journals, books, internet, etc. solely to support, elaborate, compare and extend the earlier work. Further, this work has not been submitted by me previously for any degree, nor it shall be submitted by me in the future for obtaining any degree from this University, or any other university or institution. The incorrectness of this information, if proved at any stage, shall authorities the University to cancel my degree.

Signature: Warda Rizwan **Date: 19 October, 2019**

Name of the Research Student: Warda Rizwan

Abstract

Automatic question generation in Knowledge Graphs (KG) is a novel idea. KG is the graphical representation of knowledge bases where knowledge base is world knowledge. Intelligent Tutoring System (ITS) is a computer program that uses artificial intelligence techniques to improve learning process in the knowledge domain. An ITS provides adaptive learning instruction and feedback to students in problem solving process. KG can be used in Intelligent Tutoring Systems for education purposes. In this research we address the problem of automatically generating knowledge questions from a KG with assigned difficulty level (easy and hard). Questions of this kind have ample applications, for instance, in Intelligent Tutoring Systems, to evaluate the knowledge of students in a specific domain. Automatic question generation can reduce human effort as making questions manually is resource and time consuming. To solve the problem, we propose a novel approach of generating questions automatically. To generate questions, we first select a domain for the knowledge graph; then process the KG to generate structured triple-pattern triples, which are then used to generate questions. A key challenge is estimating how to generate easy and difficult questions. To do this, we used single triple for easy questions and more than one triple for difficult questions which means that using only one triple involves two nodes and more than one triple contains more than two nodes, so when the graph is traversed for easy questions only two nodes are visited and for difficult questions more than two nodes are visited which increases the complexity.

Dedication

This Thesis is dedicated to my Parents, In Laws, Husband and beloved Daughter for their love, continuous support and encouragement regarding my goals.

Acknowledgements

Firstly, all praises to the Almighty Allah for blessing me with the strength and patience needed to complete this research.

Secondly, I would like to thank my supervisor Dr. Raja Muhammad Suleman for his endless help and guidance. He provided his support and always gave excellent advice which made my thesis a success. He was always available and all our meetings proved to be very fruitful for me. I am highly grateful to Dr. Raja Muhammad Suleman for reviewing my thesis.

Furthermore, I would like to thank my family specially my husband for helping and supporting me throughout.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Question is a linguistic expression, with the aim to seek information. Over the course of time, questions have developed a range of uses that go beyond the mere extraction of information from another party. One such use is in the field of education, where comprehension level of students can be evaluated by asking multiple questions from a selected domain. However, formulating meaningful questions demands considerable time and effort [3]. Research has focused on developing methods to automatically generate questions [17][2], these include generation of multiple-choice questions [3]. There has been some work done on Question Answering (QA) via Knowledge Graphs [5].

Question Generation (QG) can be an important feature of Intelligent Tutoring Systems. Intelligent Tutoring System (ITS) is a computer program that uses artificial intelligence techniques to improve learning process in the knowledge domain [28]. An ITS provides adaptive learning instruction and feedback to students in problem solving process. In ITS artificial Intelligence techniques provide knowledge on the basis of student learning status [30]. ITS aim to mimic human tutors in skills and behaviors that decide “what to teach” and “how to teach” on basis of student pervious knowledge [31].. ITSs have been shown to be as fruitful as human tutoring [16]. ITS model contains four components i.e. Student Module, Knowledge/Domain Module, Pedagogical/Tutor Module and a User Interface Module as shown in Figure 1.

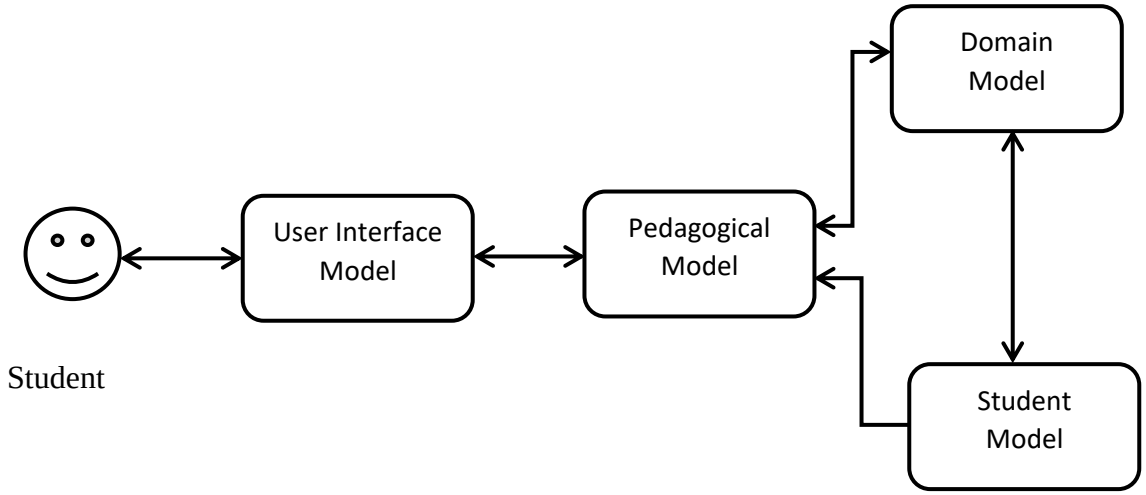

Figure 1: Intelligent Tutoring System Model

Knowledge/Domain Module represents knowledge of domain concepts, problem-solving strategies that are taught to the student. This module represents domain knowledge in a machine understandable format [10].

Student module holds knowledge about the students and their current state of learning process. This module records the student's knowledge state step-by-step when student solves a problem and the system gives an error when the student deviates from predefined model. The student knowledge state evolves when student learning process advances [10].

Pedagogical/Tutor Module is responsible for providing different teaching and tutoring strategies automatically depending on student state and history. This module manages the learning concepts and monitors the learning activities of the students [28]. This module communicates with student module to determine which concepts of domain knowledge are presented to the students on the basis of their pervious history.

User interface Module is responsible for displaying necessary information to the students giving them a path to communicate and interact with ITS. It provides domain knowledge and test results to the students [1].

In ITS domain module represents knowledge about domain in such a way that computers understand it, which is called knowledge representation. The knowledge representation makes decisions of information selection, problem selection and feedback generation. In

recent years, researchers have focused on ontology for knowledge representation when building ITSs [8].

A QG system takes a knowledge source as input along with some specifications describing the questions to be generated [29]. As for the output, it produces a reasonable number of questions. These specifications can include, for example, the format of the question and its difficulty [3]. Knowledge Graphs (KG) along with QG can be used in ITS, as it can ascertain factual and conceptual knowledge of students. This can be achieved by imparting sufficient knowledge regarding a specific domain and then challenging them with associated questions.

Some of the ITS use KG to represent their domain models. KG is the graphical representation of a knowledge base, where knowledge base is world knowledge which is general purpose and not domain specific [30]. A KG therefore, represents collection of interlinked entities which could be real-world objects having a link or a mutual relationship between them. Freebase [15], YAGO [25], Wikidata [14] and DBpedia [24] are few such examples of KG, which encompass world knowledge in wide-ranging spheres. Knowledge graphs provide world knowledge as structured (RDF) data, which uses SPARQL [26] or cypher to query that data.

The availability of QG techniques and knowledge graphs as great as these can prove useful for generating questions of a specific theme and assigned degree of difficulty, that are suitable to user requirement. By using subsequently produced questions as a medium for information, a novel use for knowledge graphs could be made. It is not surprising that knowledge graphs are increasingly used for building Question Answering systems [21] [23].

Research shows that a major part of work carried out in the field of KG, focuses primarily on generating questions automatically [13]. Along with question generation, efforts are being made on estimating difficulty level for these questions [17][2], research efforts are also being made on developing such systems that generate questions of a specific level of difficulty (i.e. low, medium and high) for its intended user.

Our work focuses on the automatic generation of questions with prescribed difficulty level. Question difficulty can be estimated by measuring the depth of KG after its traversal, which means if more nodes are involved in question generation process then the difficulty of generated question Is high. Benefits include saving human resources, generation of questions on a large scale, and evaluation of user knowledge by grading questions as per their difficulty.

1.1. Research Objective

This is an empirical research and its objective is to propose a novel approach to automatically generate questions from KG with prescribed difficulty levels

1.2. Problem Statement

Knowledge Graphs in general and Knowledge Graphs in ITSs in particular lack meta-information about the data therefore the difficulty level of automatically generated questions cannot be assessed by such systems. The system does not know the difficulty level of the generated questions unless it is pre-specified by a human author.

1.3. Proposed Solution

The goal of this research is to use the structured data of KG for generation of meaningful questions with assigned difficulty level automatically. The proposed solution would use depth analysis of a KG to generate possible difficulty levels of questions.

1.4. Research Questions

This research is guided by following research questions:

Q.1. What are the different difficulty levels of a question?

Q.2. What are the different techniques that play part in automatically generating questions from knowledge graphs?

Q.3. What are the different techniques available to determine the difficulty level of automatically generated questions?

1.5. Methodology

The research methodology that we are going to use is empirical, so our study will be empirical based where we will be generating questions from KG and then we will be experimenting on the different techniques to assess the difficulty level of a question generated from a KG. First phase of our research includes literature review leading to the research gap identification and generation of our problem statement that will be worked on. Second phase includes different techniques that we will be using for the generation of questions with estimated and assigned difficulty level.

Our objective is to come up with a question belonging to a specific topic and having a specific difficulty level. Our work will be based on using spaCy for entity recognition and data triples for query generation. By starting from entity recognition, we'll be able to generate questions that will have one correct answer in our KG and for question difficulty estimation the path generated, while traversing the KG in search of answer, will be examined and as per its depth the difficulty of question will be determined. If a greater number of nodes are visited for answer retrieval, then the difficulty of question will be considered high and vice versa.

1.6. Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 describes the background and the related work. Chapter 3 explains the technical background necessary for understanding our methodology and rest of the work. Chapter 4 contains our methodology motivation of our proposed technique, then experimental results are presents. Finally, Chapter 5 concludes our thesis.

Chapter 2

Literature Review

2.1. Intelligent Tutoring Systems

According to M. Alqahtani [1] an ITS is a computer system that uses artificial intelligence techniques to facilitate learning and teaching process. An ITS uses cognitive learning theory for representation and organization of knowledge in human memory and handle human errors. In ITS, intelligence is concerned with the representation of domain knowledge, problem generation, providing guidance and feedback to the learner at each step-in problem-solving activity as they interact with the system. ITS collects the information of particular student's performance i.e. their strengths and weaknesses, on the basis of which ITS suggest more domain knowledge to the students. The students learn from ITS in similar way as they learn from human tutor. Therefore, this type of system improves the performance of students. The goal of ITS is to provide knowledge about domain and skills about problem solving to the student independently. Computer-based training (CBT) and Computer Aided Instruction (CAI) are the earliest versions of educational system that contain sequential tutoring strategy. These systems were provided same instructions in similar way to all students at the same time, and they did not provide instructions to the specific student according to their needs. ITSs overcome this problem by taking a role of one-to- one tutor that deals with students specific needs and provides tuition according to the student profile. ITSs use artificial intelligence techniques that evaluate each student's behavior, performance and increase their knowledge. Based on the student performance ITS provides explanations, guidance, feedback and practice problems as needed.

2.2. Question Generation

Sakaguchi et al. [44] creates “fill-in-the-blanks” quizzes for language learning, by concentrating on the problem of removing words from a sentence to leave the blanks. Distractors are created for the removed words and evaluation is then carried out to check their reliability and validity. A distractor should be reliable so that it cannot be replaced with the answer. By doing so, multiple correct answers to a question can be avoided. Moreover, the distractors have to be valid. By validity it is desired that is should be “close enough” to the correct answer, in such a way that the learners are distracted regarding the actual answer. Their proposed methodology is to first find out the word to be left out, by observing at error-correction pairs taken out from a large English learner corpus and then verbs selection where a semantic mistake was made. After this, conditional probability P (we | wc) is calculated that a word wc is misused as we and compute a confusion matrix based on these probabilities. In a given sentence, the verbs appearing in the confusion matrix are marked and made blank. For generating the distractors, the authors train multiple classifiers for each target word using the error-correction pairs. These classifiers are based on the discriminative Support Vector Machine model and are trained by looking at 5-gram lemmas and their dependence types with the target word. Each trained classifier for a target word works by taking a sentence as input and giving a verb as an output, as the best distractor given the 5-word context. At last, the method is assessed in terms of its effectiveness, by conducting a user study with English speaking individuals and then comparing the ratio of appropriate distractors with two baselines. They demonstrate that their discriminative models do better than their baselines that use a generative model. Furthermore, they show the validity of their distractors by measuring high association between the performance of non-English speakers on a test produced by their system and the participant’s TOEIC2 scores.

Rus et al. [43] in their paper explained that the task of question generation is automatic generation of questions from different input sources. Raw text, some form of semantic representation or a database can be the various sources for question generation. Two main aspects of question generation have been identified by them. First one is the question’s goal and second is its importance. Furthermore, it is argued that only by looking at the

context in which the question was asked, the “goodness” of a question can be ascertained. It is therefore necessary, to find information that what was the goal of the question and to determine that what forms important regarding the current context. While carrying out examination of associated work, we found out that in practical application, many approaches that are proposed cannot be starkly categorized by a single input source, as a combination of various input sources is used in these papers.

Narendra et al. [45] from a given text, propose an end-to-end system for the automatic generation of fill-in-the-blanks questions. As an input, a text document from the Cricket domain is retrieved. A sentence with a blank and four answer options is shown as output. Out of the four answers, one is the correct answer, while the remaining three are distractors. For question generation from any given data/ document, three stages of processing are performed in their approach. A relevant and informative sentence is selected to represent the question’s sentences in the first stage. To accomplish this, the authors utilize an off the shelf extractive summarizer and use the top ten percent of output of the summarizer. In the second stage of their approach, keywords that are used as the blank in the question are selected. These keywords can be either named entities, constituents or pronouns. Furthermore, a list of observations is defined by them to help trim the list of candidate keywords, which 1A distractor is an incorrect option in a multiple choice question. In the final stage, questions distractors are generated by the researchers using an approach backed by a knowledge graph. When the selected keyword is a named entity, the knowledge graph is only involved in distractor generation. When it is a case that a named entity is not a person, their algorithm selects a fact from the knowledge graph randomly. If it is a person, the algorithm selects facts depending on the Cricket team the person plays in. By using this technique, it helps to generate distractors of players whose properties are close to the answer’s properties.

The approach introduced by Labutov et al. [46] focuses on generating high-level comprehension questions rather than just factoid questions. Their approach is distinguished from others because deep understanding of the text is not required by the

system, as crowd workers generate question templates. In this approach, questions are generated by representing the source text in ontology. The ontology is built as the Cartesian product of Freebase article categories and article section names, derived from Wikipedia. These are termed mappings category-section pairs by the authors. For example, the category Person and the section Early life form such a pair. In the next step, using such pairs from the ontology, crowd workers are asked to generate high-level templates. For the above-mentioned category-section pair a crowd worker may make the question templates who were the key influences on <Person> in their childhood? The authors build a classifier that ranks each question according to its relevance to the given text, to ensure the generated questions are high-level and relevant

A scheme that makes use of semantics provided by ontology was introduced by Al-Yahya [47] as the OntoQue engine. The author's system is supported by ontology with approximately 300 RDF triples to create multiple-choice, true/false, and fill-in-the-blank questions. OntoQue generates questions by reiterating over RDF statements that contain entities, in such a way that every statement can be curved into a single question. RDF triples that are not expressive for questions are filtered out. Fill-in-the-blank questions are generated by parting out either the subject or the object of a triple. For true/false questions either the subject or object is substituted by an entity belonging to the same class as the entity of the correct answer. Distractors for multiple-choice questions are produced by either taking into account entities that share the same class-membership as the answer or tallying all individuals in the knowledge graph and gathering all assertions where the individual is either subject or object. Furthermore, the author utilizes the rdfs:label property to access the surface form for an entity. The system was evaluated by the author, by categorizing the generated questions as good or bad, and measuring precision.

2.3. Difficulty Estimation

In this section we discuss associated work in two tricky spheres related to the estimation of difficulty in language. The first domain deals with the calculation of question difficulty in the context of public question answering services, such as StackOverflow3 and Yahoo! Answers4. The second domain deals with the prediction of reading difficulty of natural language text. Although methods of said domain do not deal categorically with questions, the discussion of this work gives understandings into the prediction models used for the approximation of difficulty in language related glitches.

Liu et al. [48] talks about the problem of estimating question difficulty in community question answering services. A competition-based approach is used, which represents question difficulty by taking the user expertise level into account. In their work they make two assumptions: First one is the difficulty of a certain question is higher than the expertise score of the one, asking the question. In the second assumption, the user's expertise, who has given the best answer, is higher than the one who asked the question and all other users who gave lower ranked answers. Question difficulty is then determined by looking at the pairwise comparisons for a "two-player" competition with one winner and one loser. Competitions can be any of the following kind:

• Competition between question and the one asking the question

• Competition between the question's asker and the one giving best answer

• Competition between the best answerer and the asker of question

• Multiple competitions between the best answerer and all other answerers

Now, the predicament of estimating difficulty of a question can be cast into the problem of acquiring relative skills of each player by observing the outcomes of the two-player competitions. If we consider the question as a participant in the competition, the difficulty of question can then be retrieved as it's skill score. Skills scores learned for all other users depict their expertise scores. To study the relative scores, the authors resort to the TrueSkill ranking model [38]. To evaluate the approach of Liu et al. [48], 300 question

pairs are sampled from StackOverflow and experts are requested to compare their relative difficulty. Then, the authors measure the correctness of their scheme as the number of correct pairwise comparisons divided by the total number of pairwise comparisons. At last, their method is compared to a Page Rank-based approach [50], where the complexity of tasks in crowdsourcing competition services is assessed. The approach models the problem as a graph, where an edge between two tasks encodes that one task is harder than the other. Then they interpret the PageRank score of each task as the difficulty measure. In Liu et al. [48], the authors' find significant performance enhancements as compared to the PageRank-based method, in terms of accuracy.

There is a body of work done in assessing reading difficulty of texts. Therefore, we place emphasis on the most related approaches in this field. Collins-Thompson et al. [51] created a method which makes use of statistical language models to measure reading difficulty. The method uses a smoothed unigram language model based on a variation of the multinomial naïve Bayes classifier. The semantic difficulty of a given text T is predicted as the probability that T was generated by a language model that represents a certain school grade level. These language models are trained from authoritative sources and educational websites that have grade levels assigned to them. Their work shows that particular words are very conclusive for a certain grade level. For example, the authors found that the words grownup, ram and planes where most representative for grade level 1, while on the other hand, words essay, literary and technology were most suggestive for grade 12. One disadvantage of their method is that it considers lexical features only and does not take into account features based on grammar. In contrast, Heilman et al. [52] present work that shows how reading difficulty estimation can be made better by taking into account a combination of lexical and grammatical features. In their approach, the authors take into consideration the relative occurrences of a set of morphologically stemmed word unigrams, which establish the lexical features. As grammatical features, this tactic computes the relative frequencies of sub-trees of syntactic parse trees up to a certain level. Using these features the researches experiment with three linear and log-linear models, namely linear regression, proportional odds model and multi-class logistic regression. These models were appraised on documents of a web corpus, where each

document had a grade level assigned to it. As a result, they found that the proportional odds model gives best results for predicting reading difficulty.

2.4. QA Systems

D. Seyler et al proposed an approach where they generated questions from KG in the form of quiz. The first step of their work was query generation, then the next step was to estimate the difficulty, and finally after these two steps they verbalized their queries. Their approach addresses the challenges inherent to this problem of generating quiz-style knowledge questions from knowledge graphs, most importantly estimating the difficulty of generated questions. Suitable features were engineered by them along with training a model of question difficulty by using the past data from the Jeopardy! Quiz show [2].

I. V. Serban et al, past work has focused on generating questions from Freebase KB. Question generation has been framed as taking a fact from freebase, shown as a triple having a subject, a relationship and an object, which was converted into a question about the subject, and the correct answer, was the object [9].

R. S. Mittal used ConceptNet5.4 as a common-sense knowledge base (KB) and generated a diverse set of MCQs for assessing conceptual understanding of a word [11].

Q. Guo et al, work has been done on the introduction of a system known as Questimator that automatically generates quizzes having multiple choice questions provided a topic from a knowledge base like Wikipedia [12].

S. Indurthi et al research has proposed a technique for creating QA sets for a specific entity utilizing a KG. Additionally, an RNN based methodology has been proposed for producing natural language questions from an input keyword sequence [13].

A. Abujabal et al presented NEQA, a continuous learning paradigm for question answering over knowledge bases. Templates mapping syntactic structures are automatically learned by NEQA when offline, to semantic ones This is done through a small number of training question-answer pairs. Once it is employed, constant learning is activated on all those cases where templates are not adequate. Using a semantic similarity function between questions and by judicious invocation of non-expert user feedback, NEQA learns new templates that capture previously-unseen syntactic structures. This way, NEQA progressively extends its template repository. NEQA periodically re-trains its underlying models, letting it adapt to the language used after deployment. Their experiments demonstrated NEQA's feasibility, with sound improvement in answering quality over time, and the ability to answer questions from new domains [6].

H. Li et al presented two approaches to tackle the problem of how to match relations in QA systems over knowledge bases. The first approach attempts to learn the soft match directly between the relations and question from the training data while using neural networks. The second approach supplements the relation name with natural language support sentences which are generated from Wikipedia, which offer additi onal matches with the question. Experiments on the WebQuestions dataset demonstrate that both of their approaches improve the precision of relation matching of a preceding state-of-the-art. Their further analysis discloses the high quality of support sentences and recommends the rich potential of support sentences in question answering and semantic parsing tasks [7].

Chapter 3

Technical Background

In this chapter we describe the important background technical knowledge which is necessary for better understanding of concepts used in this thesis.

3.1. Knowledge Graph

The knowledge graph is a collection of interlinked entities which are real-world objects, situations, events, or abstract concepts. Knowledge Graph is the graphical representation of a knowledge base, where knowledge base is world knowledge which is general purpose and not domain specific.

Key Characteristics

The knowledge graph has different data management system aspects. We can see it as an explicit type of:

database, as we can query it using structured queries;

graph, as it can be analyzed just like other network data structure;

Knowledge base, as its data has formal semantics, by interpreting the data mnew facts cab be inferred

3.2. RDF

RDF stands for Resource Description Framework. To describe the properties of resources RDF is used. The data model which is provided by RDF describes these properties in the form of subject-predicate-object triples. Resource is the subject which is described. The object can be fixed value or any other resource which is

called literal. The predicate shows the sort of the connection between the subject and the item and is spoken to utilizing a property (e.g., the property rdf:type demonstrates that an asset is an occasion of a class) []. A lot of these triples structure a marked, coordinated multigraph that can be questioned utilizing the SPARQL inquiry language.

3.3. SPARQL

The SPARQL Protocol and Query Language (SPARQL) can be utilized to recover or control information in the RDF diagram [32]. The aftereffects of SPARQL inquiries can be results sets or RDF diagrams." Conjunctions are communicated by the utilization of regular factors and are indicated by a main question mark. Disjunctions give the ability to recover a coordinating subgraph if in any event one of numerous diagram examples matches. In SPARQL a disjunction is communicated utilizing the UNION catchphrase.

3.4. Knowledge Bases

A knowledge base is a, centrally reachable, aggregation of information. For example, a public library, a domain specific database or an online encyclopedia, , can all be generally regarded as knowledge bases [55]. In recent years the term has been used especially to refer to a database that stores information in an ontological representation. These knowledge bases store knowledge about classes and their relations and combine them with instance-level knowledge. In addition to instance-class affiliations, they store information about the relations between entities.

Knowledge bases fill different needs crosswise over different domains. They can contain lexical data (e.g., Wordnet [33]), which is used in the field of semantics. They can contain good judgment learning (e.g., WebChild [34]), which can be utilized for thinking and question replying in artificial knowledge. Knowledge bases

that contain exceptionally wide learning and are not limited to a specific space are called broadly useful information bases. Models for these frameworks include YAGO, Freebase, DBpedia and numerous others.

3.5. Neo4j

Neo4j is a graph database management system which is created by Neo4j, Inc. Portrayed by its designers as an ACID-agreeable value-based database with local chart stockpiling and handling, Neo4j is the most well-known diagram database as per DB-Engines positioning, and the 22nd most prevalent database by and large. Neo4j is accessible in a GPL3-authorized open-source "network version", with online reinforcement and high accessibility expansions authorized under a shut source business permit. Neo likewise licenses Neo4j with these augmentations under shut source business terms. Neo4j is actualized in Java and open from programming written in different dialects utilizing the Cipher Query Language through a value-based HTTP endpoint, or through the binary "bolt" protocol

3.6. Cypher Query Language

Cypher is a declarative graph query language that allows for expressive and efficient querying and updating of a property graph. It is a moderately basic yet exceptionally amazing language. It is also able to demonstrate complex database questions without much of a stress. This enables clients to concentrate on their area as opposed to becoming mixed up in database outflow [39].

Cypher was to a great extent a development of Andrés Taylor while working for Neo4j. It was initially proposed to be utilized with the graph database Neo4j, later it was operated through the openCypher venture [40].

3.7. Python

Guido van Rossum designed and released Python in 1991, which is a broadly useful programming language. Python's plan theory underscores code clarity with its eminent utilization of noteworthy whitespace. Its language intends to enable developers to compose clear, consistent code for little and huge scale ventures. Python underpins numerous programming ideal models, including procedural, object-arranged, and practical programming. It is progressively composed and trash gathered. Python is frequently depicted as a "batteries included" language because of its far reaching standard library.

3.8. NER

NER stands for Named-entity recognition which aids in statistic extraction that tries to arrange and locate named elements makes reference to in unstructured content into pre-characterized classes, for example, financial qualities, associations, the individual names, rates and so on. Most research on NER frameworks has been organized as taking an unannotated square of content, for example, this one: In 2015, 600 portions of Wheat Crop were purchased by Jack. Furthermore, delivering ann annotated block of content that features the names of substances:

[Jack]Person purchased 600 portions of [Wheat Corp.] Organization in [20115] Time.

In this model, an individual name comprising of one token, a two-token organization name and a temporal expression have been identified.

3.9. SpaCy

SpaCy is an open source library for advanced Natural Language Processing in Cython and Python. The library is issued under the MIT permit and at present offers factual neural system models for English, Portuguese, French, German, Spanish, Italian, Dutch and multi-language NER, just as tokenization for different dialects.

SpaCy's named entity recognition trained on the OntoNotes 5 corpus supports the following entity types as shown in Table 1:

TYPE	DESCRIPTION
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.
LOC	Non-GPE locations, mountain ranges, bodies of water.
PRODUCT	Objects, vehicles, foods, etc. (Not services.)
EVENT	Named hurricanes, battles, wars, sports events, etc.
WORK_OF_ART	Titles of books, movies songs, etc.
LAW	Named documents made into laws.
LANGUAGE	Any named language.
DATE	Absolute or relative dates or periods.
TIME	Times smaller than a day.
PERCENT	Percentage, including ”%“.
MONEY	Monetary values, including unit.
QUANTITY	Measurements, as of weight or distance.
ORDINAL	“first”, “second”, etc.
CARDINAL	Numerals that do not fall under another type.

Table 1: Entity Types Recognized by Spacy

3.10 . The RDF and Labeled Property Graph Models

Neo4j uses Labeled Property Graph (LPG) models. Comparing these two models, the RDF model is about exchanging data while the labeled property graph is mainly about storage and querying. As we know that there are two components of a graph:

nodes and the edges. In LPG, nodes have a unique ID along with a set of key-value pairs. Likewise, edges which are known as relationships, contain an ID. In labeled property graph both the nodes and relationships have an internal structure i.e. set of key-value pairs, which differentiates LPG model from RDF model.

The RDF model consists of triples, consisting three elements which include two vertices connected by an edge. It's known as subject-predicate-object. Here subject and object are nodes while predicate represents an edge.

Chapter 4

Methodology

In this chapter we will explain our methodology of building our own small knowledge graph and then using it for automatic question generation. Section 4.1 shows why we haven't used big KGs like YAGO, DBpedia, BabelNet and Freebase, In Section 4.2 we will explain how Neo4j was used for a small KG, section 4.3 shows our methods and strategies for generating questions. In section 4.4 we explain our findings and results.

4.1 Huge Knowledge Bases

For our experimentation we preferred to make a small KG of our own instead of using other ones like YAGO (Yet Another Great Ontology) and DBpedia because of their size and complexity. YAGO is a gigantic semantic knowledge base, derived from Wikipedia WordNet and GeoNames. Presently, YAGO contains knowledge of more than 10 million entities (like persons, organizations, cities, etc.) and comprises of more than 120 million facts about these entities. YAGO syndicates the clean taxonomy of WordNet with the fullness of the Wikipedia category system, assigning the entities to more than 350,000 classes [25].

DBpedia is a cross-domain ontology, which has been manually created based on the most commonly used infoboxes within Wikipedia. The ontology at present covers 685 classes and are described by 2,795 different properties [24].

Freebase was a huge collective knowledge base composed primarily of its members. This is an online collection of structured data from numerous sources, including individual wiki contributions submitted by the user [37].

The Knowledge Graph announced on 16 December 2014 that it would shut down Freebase for the six months that followed and assist to transfer information from Freebase to Wikidata [36]. Google formally announced the Knowledge Graph API on 16 December 2015, which is intended to replace the Freebase API. On 2 May 2016, Freebase.com was formally shut down [38].

4.2 Building a small knowledge Graph using Neo4j

. Our first step to making a KG was to select software and a domain to work with. Neo4j our chosen domain was ‘Movies’ as Neo4j has an example KG of this domain.

Following are some of the advantages of Neo4j [56] .

- Neo4j has a strong data model that can be readily modified by applications.
- Neo4j offers real-time outcomes.
- Neo4j is extremely available with transactional guarantees for big business real-time apps.
- It allows you to represent semi-structured and linked information readily.
- We can not only depict but can also quickly recover linked data from other databases by using Neo4j.
- Neo4j represents the chart visually with cypher query language. The commands in this language are readable and simple to understand by humans.
- Neo4j uses a graphical model where there are nodes (entities) and these nodes are linked with edges (relationships). Nodes and relationships store information in property key value pairs.

Neo4j Graph Database comprises −

- Nodes
- Labels
- Properties
- Relationships
- Data Browser

Node

Node is a central component of a Graph having properties with key-value pairs as shown in Figure 4.1.

Figure 2: Person Node

Node Name = "Person" having a set of properties.

Relationships

Relationship, known as an edge, is a connected between two nodes as shown in the following Figure 3.

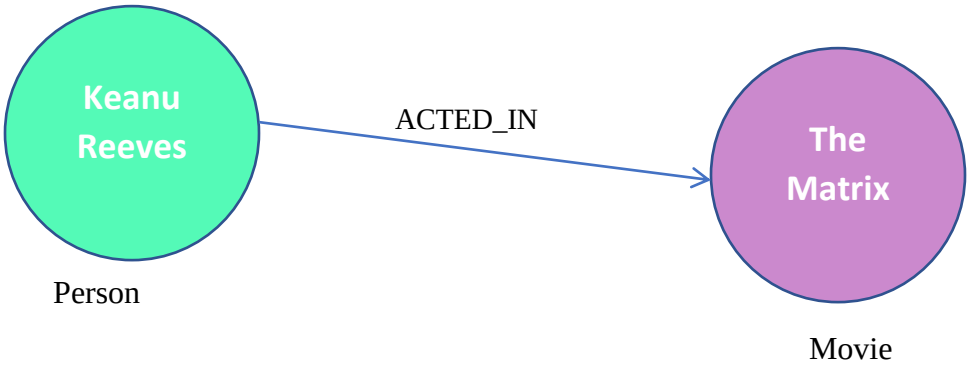

Figure 3: Graph Showing Node Relationship

Here, Keanu Reeves and The Matrix are two nodes. "ACTED_IN" is an edge/relationship between these two.

Thhe arrow mark from Keanu Reeves and The Matrix, this relationship describes:

‘Keanu Reeves’ ACTED_IN ‘The Matrix’.

Here, "Keanu Reeves" is the starting node, and "The Matrix” an end node.

The arrow mark represents a relationship from "Person" node to "Movie" node, this relationship is known as an "Incoming Relationship" to "Movie" Node and "Outgoing Relationship" to "Person" node.

Just like nodes, relationships can hold properties as key-value pairs.

Properties

Property is a key-value pair to describe Graph Nodes and Relationships.

Key = Value

Where Key is a String and Value may be represented using any Neo4j Data types

Labels

Labels are like a collective name to a set of nodes or relationships. There could be more than one label for a relationship or node. New labels can be assigned to already present nodes or relationships. Labels can be eliminated from the existing nodes or relationships.

Figure 3 shows that there are two nodes.

The Left side node has Label: "Person" and the right side node has a Label: "Movie".

Their Relationship also has a Label: "ACTED_IN".

A node can have multiple labels. Labels can be separated for the node with a colon i.e. “ : ”.

CREATE (Keanu Reeves:person:actor)

Next step was to populate this KG, so we added some more data, to the example ‘Movies’ KG, like Countries where people were born. Figure 4 shows this relationship and entities. Cypher CREATE query is used to generate the KG and Figure 5 shows a part of cypher queries written for this purpose.

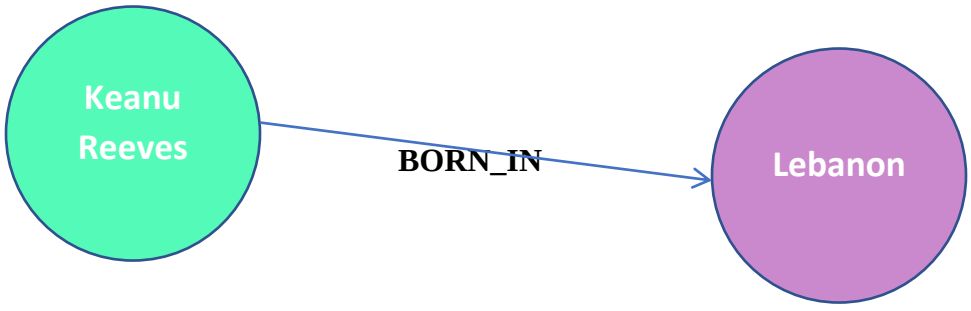

Figure 4: Single Triple Graph

```
CREATE (TheMatrix:Movie {title:'The Matrix', released:1999})
CREATE (Keanu:Person {name:'Keanu Reeves', born:1964})
CREATE (Leban:Place {name:'Lebanon'})
CREATE (Carrie:Person {name:'Carrie-Anne Moss', born:1967})
CREATE (Can:Place {name:'Canada'})
CREATE (Laurence:Person {name:'Laurence Fishburne', born:1961})
CREATE (Usa:Place {name:'United States'})
CREATE (Hugo:Person {name:'Hugo Weaving', born:1960})
CREATE (Nig:Place {name:'Nigeria'})
CREATE (LillyW:Person {name:'Lilly Wachowski', born:1967})
CREATE (LanaW:Person {name:'Lana Wachowski', born:1965})
CREATE (JoelS:Person {name:'Joel Silver', born:1952})
CREATE
  (Keanu)-[:ACTED_IN {roles:['Neo']}]->(TheMatrix),
  (Keanu)-[:BORN_IN] ->(Leban),
  (Carrie)-[:ACTED_IN {roles:['Trinity']}]->(TheMatrix),
  (Carrie)-[:BORN_IN]->(Can),
  (Laurence)-[:ACTED_IN {roles:['Morpheus']}]->(TheMatrix),
  (Laurence)-[:BORN_IN]->(Usa),
  (Hugo)-[:ACTED_IN {roles:['Agent Smith']}]->(TheMatrix),
  (Hugo)-[:BORN_IN]->(Nig),
  (LillyW)-[:DIRECTED]->(TheMatrix),
  (LanaW)-[:DIRECTED]->(TheMatrix),
  (JoelS)-[:PRODUCED]->(TheMatrix)
```

Figure 5: Cypher queries for generating KG

Structure of Knowledge Graph:

As we have discussed in section 3.1 that a KG contains data in the form of nodes and edges, following is the database information of our KG

Node Labels

Our KG has a total of 175 nodes and there are 5 node labels which are as follows:

- Movie
- Person
- Place
- Date

- Money

Relationship Types

There are 257 edges/relations and have the following types:

- ACTED_IN
- ACTED_AS
- WORKED_AS
- BORN_IN
- DIRECTED
- FOLLOWS
- PRODUCED
- REVIEWED
- WROTE
- SECOND_SEQUEL
- THIRD_SEQUEL
- FILMED_AT
- EARNED
- WON_AN

Property Keys

- born
- name
- rating
- released
- roles

Figure 6 shows a part of generated Knowledge Graph. Here the green colour shows persons, red colour shows movies and purple colour shows places. These are the entities of our KG while arrows show the relationship between these entities, such as Keanu Reeves acted_in The Matrix.

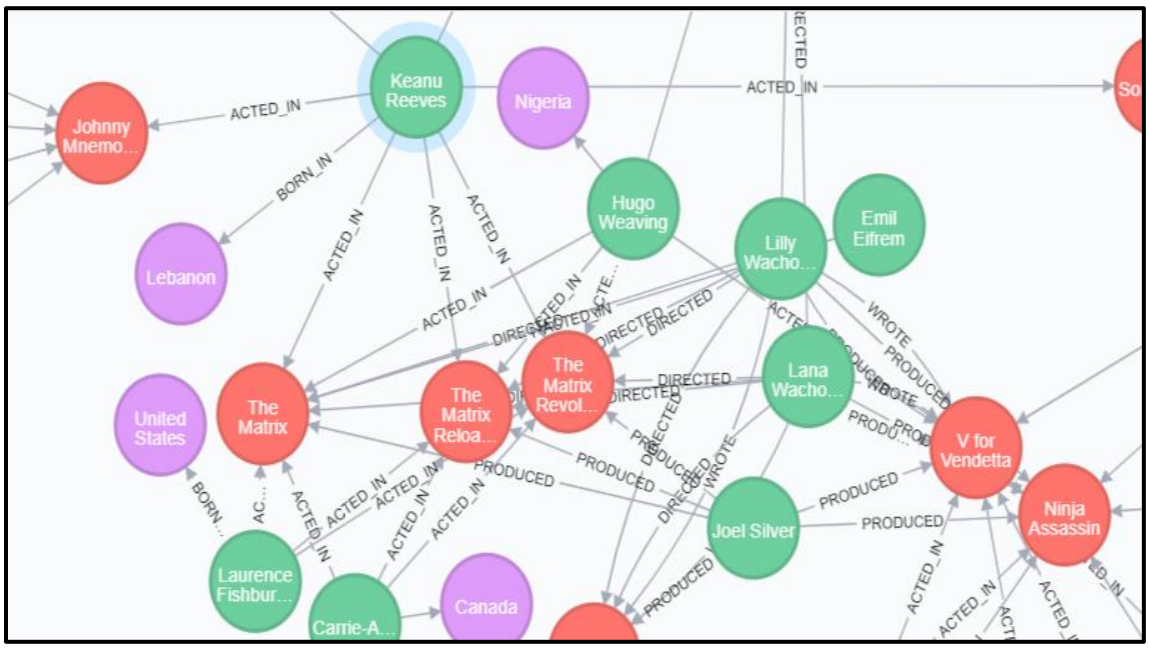

Figure 6: Generated Knowledge Graph

4.3. Generating Questions from Knowledge Graph

Our approach for generating questions is divided in two parts, first is Entity Recognition and second is Question Generation using triples.

4.3.1. Entity Recognition

For question generation we first need to recognize the entities of our KG. By default the entities are not differentiated as person, location and others in KG so we used Python to create a model for our system. We used spaCy as it is an open source advanced Natural Language Processing library in Python. We trained spaCy NER (named entity recognizer) for entity recognition by providing it the entities as training data and created a model. Following are the steps that we followed:

1. We created an empty model using spacy.blank with the ID of our language i.e. English and added the entity recognizer to the pipeline

```
# create blank spacy model
nlp = spacy.blank('en')
print('created blank model')

if 'ner' not in nlp.pipe_names:
    ner = nlp.create_pipe('ner')
    nlp.add_pipe(ner, last=True)
```

2. Next step was to go through the words of the input and update the model, which is done by calling nlp.update. A prediction is made at each word. Then annotations are consulted to see whether it was right.

3. Finally the trained model is saved using nlp.to_disk.

```
# save trained model
if not os.path.exists(output_dir):
    os.mkdir(output_dir)

nlp.to_disk(output_dir)
```

4. At the end the model is tested for ensuring the correct recognition of entities.

```
# test NER on training data. Comment below code if training data is huge.
for text, _ in training_data:
    doc = nlp(text)
    print(text)
    print('Entities', [(ent.text, ent.label_) for ent in doc.ents])
    print("")
```

Following are the test results and Table 4.2 shows some of the entities recognized by spaCy.

Keanu Reeves acted in The Matrix.

Entities [('Keanu Reeves', 'person'), ('The Matrix', 'work_of_art')]

Keanu Reeves was born in Lebanon.

Entities [('Keanu Reeves', 'person'), ('Lebanon', 'location')]

Person	*Keanu Reeves, Lilly Wachowski, Joel Silver, Neo*
Location	*Lebanon, Canada, USA*
Date	*1999, 2001, 2005*
Work_of_art	*The Matrix, The Matrix Reloaded, Top Gun*
Money	*$430 million*

Table 2: Recognized Entities

4.3.2. Question Generation

Once our entities are recognized they can be used for question generation. We experimented with two levels of question difficulty i.e. easy and hard and created a total of 40 questions, in which 20 were easy and 20 hard level difficulty. For easy questions our system requires only one triple and generates a question out of it and for hard level questions it takes two triples. Once the model is created we can then generate questions. Mapping for easy question generation is shown in Figure 7. The mapping is done according to the following rule base:

A triple is expressed as <subject, predicate, object>

1. *(Keanu Reeves, born_in, Lebanon)*

   {

   If(subject= person && object=location)

   {

   Where was subject (Keanu Reeves) born?

   }

}

2. *(Keanu Reeves, acted_in, The Matrix)*

   (Lilly Wachowski, directed, The Matrix)

   (Lilly Wachowski, wrote, The Matrix)

   (Joel Silver, produced, The Matrix)

   {

   If(subject= person && object= Work_of_Art)

   {

Which movie subject (person) predicate (acted_in/directed/produced/wrote)?

Who predicate (acted_in/directed/produced/wrote) object (movie name)?

}

}

3. *(The Matrix, released_in, 1999)*

{

If(subject= Work_of_Art && object=date)

{

When was subject (movie) released?

}

}

```
# this mapping returns template for the question.
mapping = {
    'person,location': {
        'born_in': 'Where was {_subject} born'
    },
    'person,date': {
        'born_on': 'When was {_subject} born'
    },
    'person,work_of_art': {
        'acted_in': 'Who {_predicate} {_object}',
        'directed': 'Who {_predicate} {_object}',
        'produced': 'Who {_predicate} {_object}',
        'wrote': 'Who {_predicate} {_object}',
    },
    'work_of_art,date': {
        'released_in': 'When was {_subject} released'
    },

    'work_of_art,money': {
        'earned': 'How much {_subject} earned',
        'costed': 'How much {_subject} costed'
    },
    'work_of_art,work_of_art': {
        'second_sequel': 'What was the second sequel to {_object}',
        'third_sequel': 'What was the third sequel to {_object}',
        'fourth_sequel': 'What was the fourth sequel to {_object}',
        'fifth_sequel': 'What was the fifth sequel to {_object}'
    },
    'work_of_art,location': {
        'filmed_at': 'Where was {_subject} filmed'
    }
```

Figure 7: Mapping for Question generation

Our system generating easy English Language Questions.

```
# main function to generate questions from edges
def generate_question_from_edge(edge, ner, default_subject_entity='person', default_object_entity='location'):
    _subject, _predicate, _object = edge

    tuple_sentence = ' '.join(edge)
    doc = ner(tuple_sentence)

    subject_entity = default_subject_entity
    object_entity = default_object_entity

    if len(doc.ents) > 0:
        if doc.ents[0].start_char == 0:
            subject_entity = doc.ents[0].label_
        else:
            object_entity = doc.ents[0].label_

    if len(doc.ents) > 1:
        subject_entity = doc.ents[0].label_
        object_entity = doc.ents[1].label_

    formated_predicate = ' '.join(_predicate.split('_'))

    mapping_key = '{},{}'.format(subject_entity, object_entity)
    question_template = mapping.get(mapping_key)
    if question_template:
        question_template = question_template.get(_predicate)
        if question_template:
            question = question_template.format(_subject=_subject,
                                                _predicate=formated_predicate,
```

System generating Difficult Questions

```
def generate_question_from_two_edges(edge_a, edge_b, ner, default_subject_entity='person',
                                     default_object_entity='location'):
    _subject_a, _predicate_a, _object_a = edge_a
    _subject_b, _predicate_b, _object_b = edge_b

    tuple_sentence_a = ' '.join(edge_a)
    tuple_sentence_b = ' '.join(edge_b)
    doc_a = ner(tuple_sentence_a)
    doc_b = ner(tuple_sentence_b)

    subject_entity_a = default_subject_entity
    object_entity_a = default_object_entity
    subject_entity_b = default_subject_entity
    object_entity_b = default_object_entity

    if len(doc_a.ents) > 0:
        if doc_a.ents[0].start_char == 0:
            subject_entity_a = doc_a.ents[0].label_
        else:
            object_entity_a = doc_a.ents[0].label_

    if len(doc_a.ents) > 1:
        subject_entity_a = doc_a.ents[0].label_
        object_entity_a = doc_a.ents[1].label_

    if len(doc_b.ents) > 0:
        if doc_b.ents[0].start_char == 0:
            subject_entity_b = doc_b.ents[0].label_
        else:
            object_entity_b = doc_b.ents[0].label_
```

Following are a few questions generated by our system using a single triple.

1. *('Keanu Reeves', 'born_in', 'Lebanon')*

   Where was Keanu Reeves born?

2. *('Keanu Reeves', 'born_on', 'September 6, 1964')*

   When was Keanu Reeves born?

3. *('Keanu Reeves', 'acted_in', 'The Matrix')*

   Who acted in The Matrix?

4. *('The Matrix', 'released_in', '1999')*

   When was The Matrix released?

5. *('Robert Zemeckis', 'directed', 'The Polar Express')*

   Who directed The Polar Express?

6. *(The Matrix Reloaded, second_sequel, The Matrix)*

   What was the second sequel to The Matrix?

7. *(The Matrix Revolutions, third_sequel, The Matrix)*

   What was the third sequel to The Matrix?

8. (*Jessica Thompson, reviewed, The Replacements)*

   Who reviewed The Replacements?

9. *('The Matrix', 'filmed_at', 'Sydney')*

Where was The Matrix filmed?

10. *('The Matrix', 'earned',* '$460 million')

How much The Matrix earned?

For hard difficulty questions our system expects two triples. At first the system selects a triple and then compares it with another triple to see whether their subjects are same, if the two triples have different subjects and objects, the system discards that triple and selects another triple for matching, and this process continues until two triples having same subject are selected, then a question is generated from those triples. Following are some of the questions:

- Triples having same subject but different object.

1. *<Keanu Reeves, ACTED_IN, The Matrix>*
   <Keanu Reeves, ACTED_AS, Neo>

   Who acted in The Matrix as Neo?

2. *<Carrie-Anne Moss, ACTED_AS, Trinity>*
   <Carrie-Anne Moss, ACTED_IN, The Matrix>

   Who acted as Trinity in The Matrix?

3. *<Slumdog Millionaire, released_in, 2008>*
   <Slumdog Millionaire, won_an, Oscar>

   Which movie released in 2008 won an Oscar?

4. *<Tom Cruise, was_born_in, 1962>*
   <Tom Cruise, acted_in, A Few Good Men>

   Who was born in 1962 acted in A few good men?

5. *<The Matrix Revolutions, third_sequel, The Matrix>*
   <The Matrix Revolutions, released_in, 2003>

   When was the third sequel to The Matrix released?

6. *<The Matrix Reloaded, second_sequel, The Matrix>*
   <The Matrix Reloaded, filmed_at, Sydney>

   Where was the second sequel to The Matrix filmed?

7. *< Joel Silver, directed, The Matrix>*
   < Joel Silver, directed, Die Hard>

   Who directed The Matrix and Die Hard?

4.3.3. Graph Traversal for complexity

The most vital graph task is to visit nodes and relationships in a methodical way – this is called traversing a graph. Traversal means going through one node to another using predecessor and successor operations in a sorted order. Although this sounds

simple, because the sorted order is logical, the next hop is determined by a node's logical predecessor or successor and *not* by its physical nearness [41]. Complexity increases as more nodes are factored in during the traversal.

There are two basic graph search algorithms: depth-first and breadth-first.

Depth- First algorithm moves from a starting node to certain end node before reiterating the search down another path from the same starting node until and unless the query is answered. The most basic level of depth-first is an *uninformed* search, where a path is searched by the algorithm until it reaches the end of the graph, then goes into reverse to the start node and tries a different path. On the other hand, dealing with semantically rich graph databases allows for *informed* searches, which terminates the search early if nodes with no compatible outgoing relationships are found. As a result, informed searches tend to have lower execution times. Cypher queries graph traversals generally perform informed searches.

On the contrary Breadth-first search algorithms traverse the graph one layer at a time. They initiate with nodes one level deep away from the start node, then goes to depth two, then depth three, and so on until the whole graph has been traversed.

After question generation comes the task of measuring question difficulty, with cypher query we can find the complexity of our questions. Using cypher we can calculate the shortest path between two nodes. Neo4j uses a fast-bidirectional breadth-first search (BFS) algorithm which is always positive to return the right answer. The time complexity of BFS is **O(V + E)**, where V is the number of nodes and E is the number of edges. For BFS to work on our questions we first need to convert them into cypher queries. It is possible to convert an English language question into Cypher using a neural network, and then run that query against a Neo4j graph database to produce an answer, Figure 8. shows the flow of conversion [53]. For our experiment we converted our questions to cypher query manually as our data was not huge.

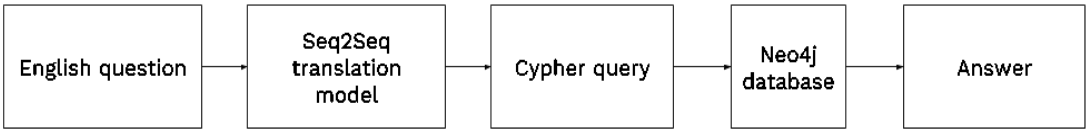

Figure 8: English to Cypher Query Conversion

Once our English language question is converted into cypher query we can get the graph of our question. Figure 9 shows the results after running the following cypher query:

Triples: *<Keanu Reeves, ACTED_IN, The Matrix>*
<Keanu Reeves, ACTED_AS, Neo>

Question: *Who acted as Neo in The Matrix?*

Cypher Query for shortest path: MATCH p=shortestPath(

(TheMatrix: Movie {title: "The Matrix"}) -[*]- (neoo: Role {name: "Neo"})

)

RETURN p

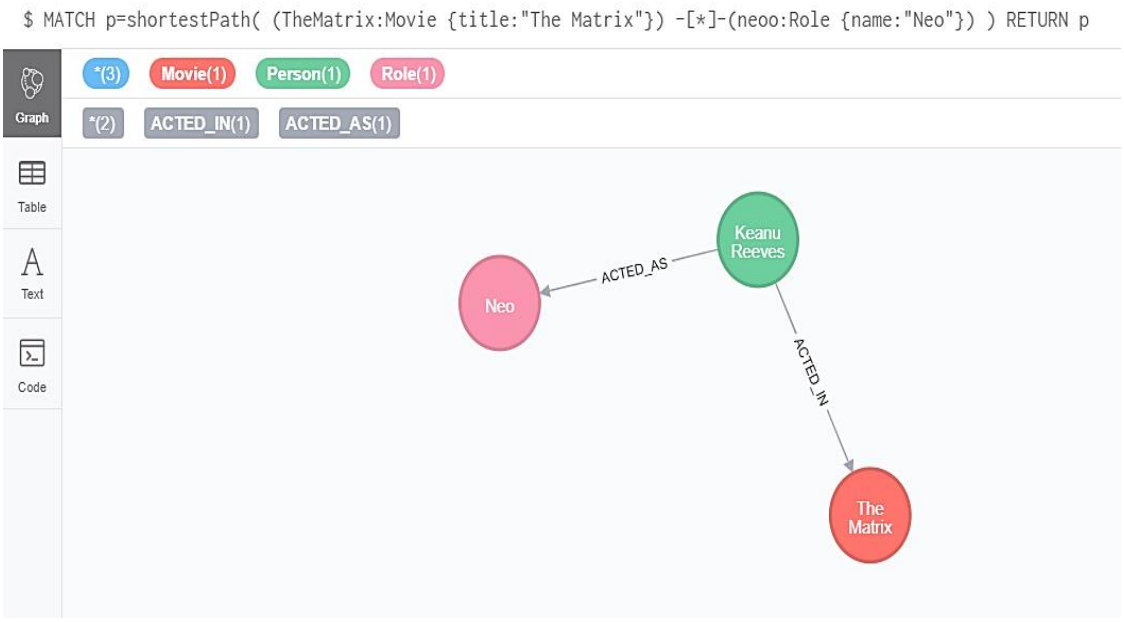

Figure 9: Results of Cypher Query

Here in Figure 9, the results are explained as follows:

At the top the query is shown and below it the details of graph are given. Number of nodes traversed is shown as *(3) in blue colour, which tells the total number of nodes traversed after the query is run. After that node labels are given as Movie (1), Person (1) and Role (1). We can also see the number of relations as *(2), there are total 2 relations one Is ACTED_AS and the other is ACTED_IN. As more than two nodes are traversed so the difficulty of this question is high. Here the BFS time complexity of traversal is O(2+3).

4.4. Results of Experimentation

Our research is empirical, in which we experimented question generation with prescribed difficulty level. Our approach was to use the structured data of KG and generate questions from it with two levels of difficulty i.e. easy and hard. The difficulty level was to be determined by measuring the depth of KG through its traversal. If more

nodes are involved in the generation of a question then the difficulty level is hard and if few nodes are involved then the difficulty level is easy.

Our system generated questions with two levels of difficulty using the data of KG as triples. For generation of easy difficulty level questions, the system used a single triple as shown in Figure 10, which means only two nodes i.e. 'Carrie-Anne Moss' and 'The Matrix' were traversed and they have a single relationship between them i.e. ACTED_AS. So according to graph traversal two nodes consisting one relationship were traversed during generation of easy question which means the path traversed for finding the answer was a simple path with complexity as O(1+2).

Graphical Results of Easy Questions:

Question: ***Who acted in The Matrix?***

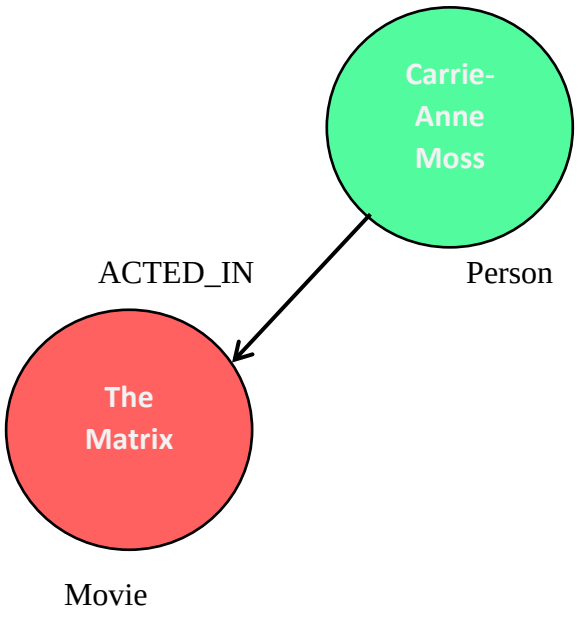

Figure 10: Result Graph Representing Easy Question

Question: Where was Keanu Reeves born?

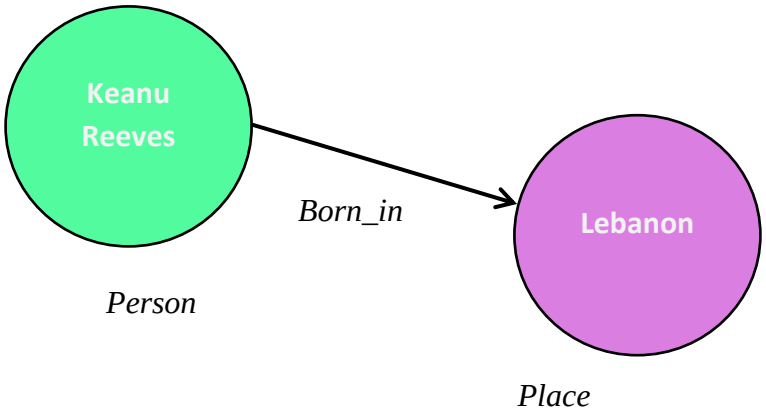

Figure 11: **Where was Keanu Reeves born?**

Question: When was The Matrix released?

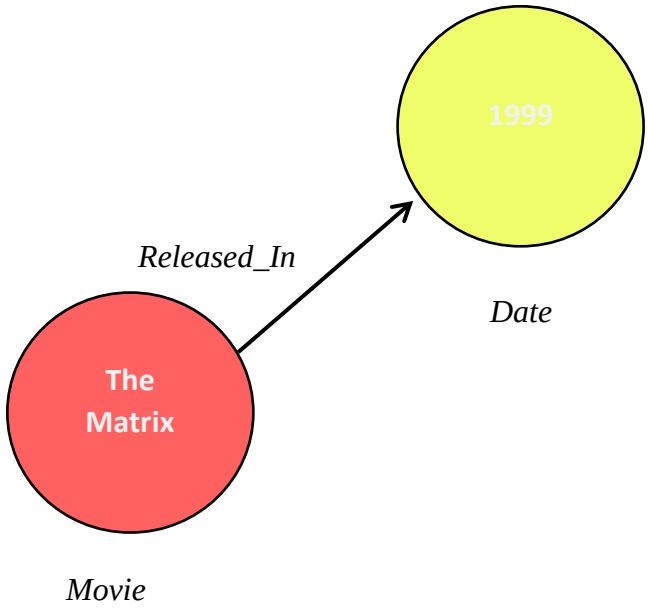

Figure 12: **When was The Matrix released?**

Question: Who directed The Polar Express?

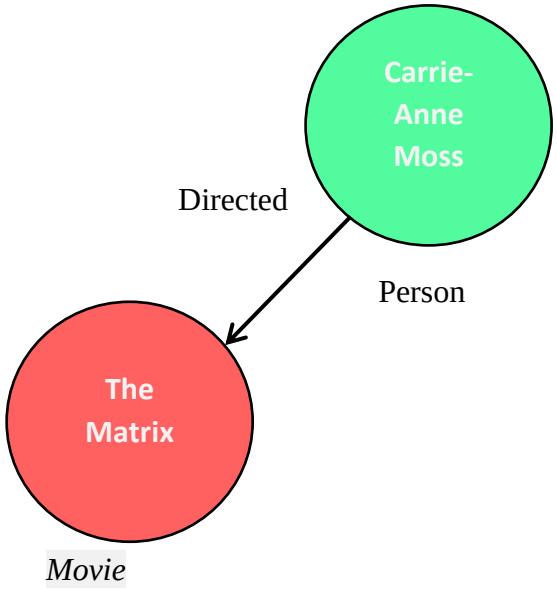

Figure 13: **Who directed The Polar Express?**

Question: What was the second sequel to The Matrix?

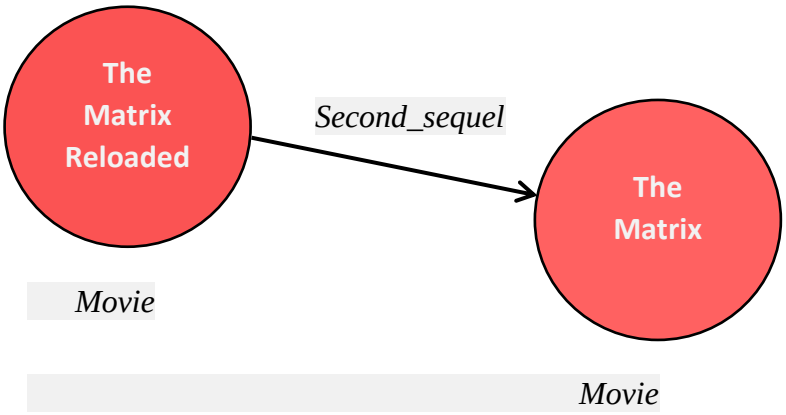

Figure 14: **What was the second sequel to The Matrix?**

Question: Where was The Matrix filmed?

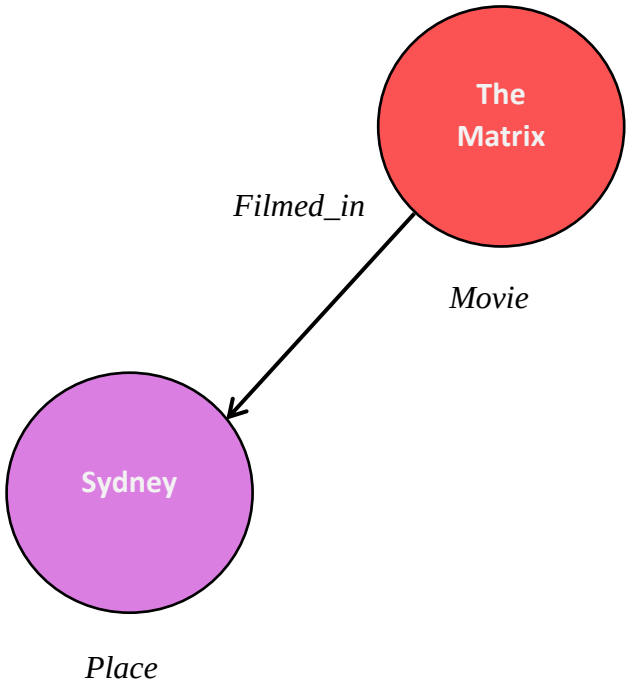

Figure 15: ***Where was The Matrix filmed?***

Question: How much The Matrix earned?

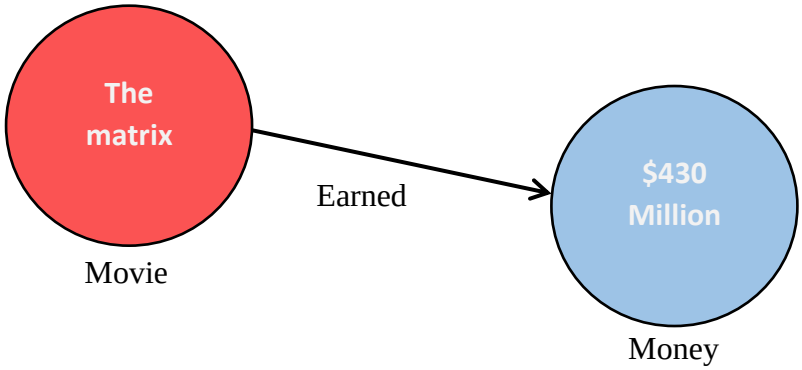

Figure 16: ***How much The Matrix earned?***

Graphical Results of Difficult Questions:

Generation of hard difficulty level questions was done by using more than one triple as shown in Figure 17, which shows that more than two nodes were involved and traversed during answer retrieval. As there are three nodes involved along with two relationships we get BFS traversal complexity as O(2+3). This shows that graph traversal for difficult questions is complex as compared to easy ones hence the difficulty is high.

Question: Who acted as Neo in The Matrix?

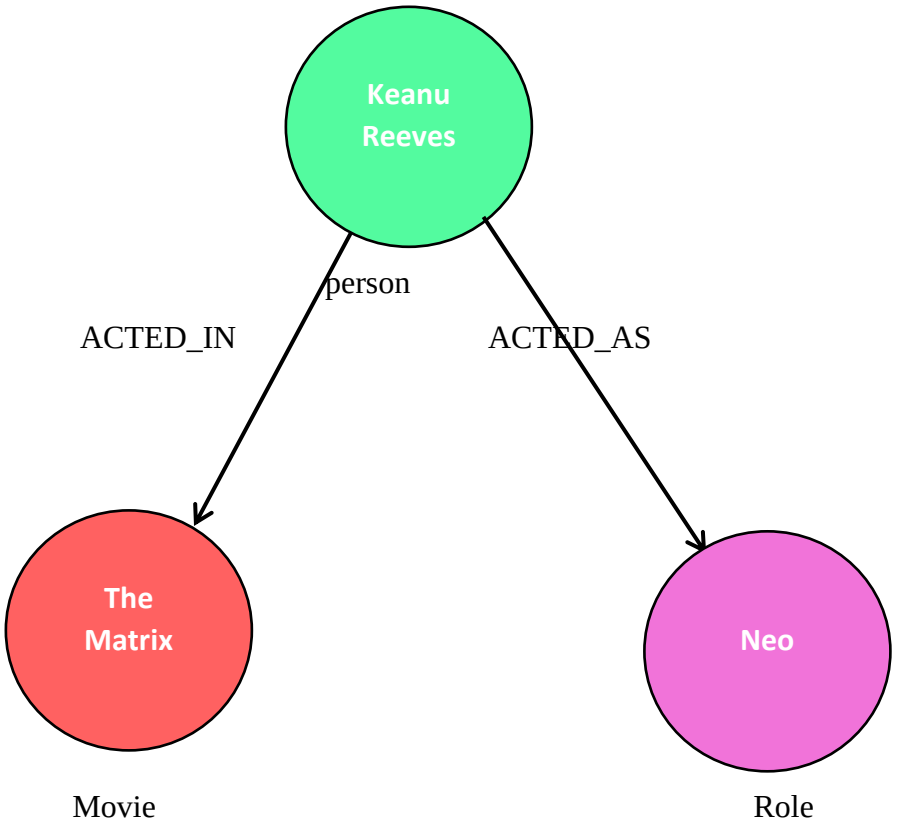

Figure 17: ***Who acted as Neo in The Matrix?***

Question: Which actor born in 1962 acted in A few good men?

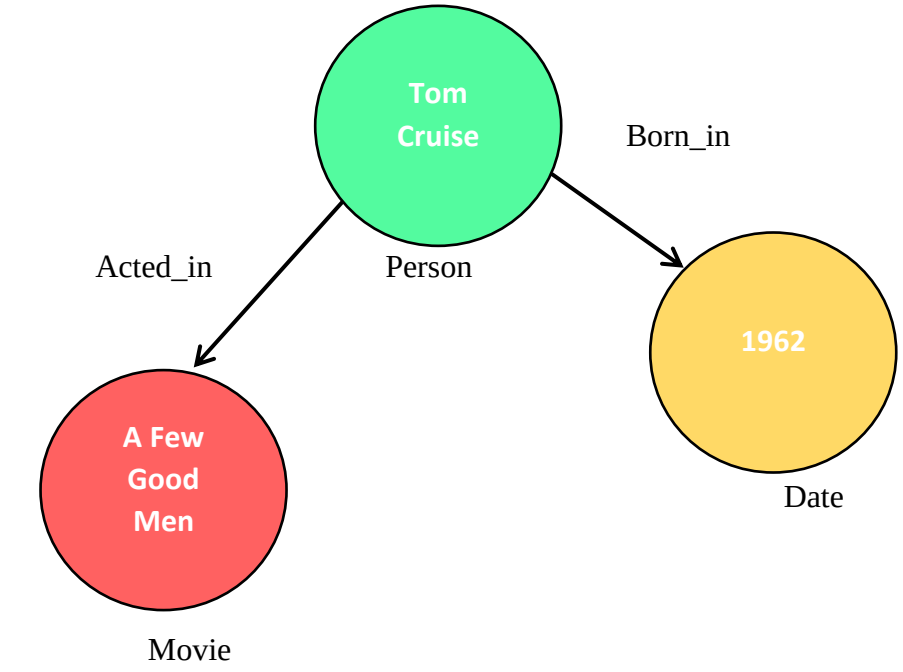

Figure 18: ***Which actor born in 1962 acted in A few good men?***

Question: Which movie released in 2008 won an Oscar?

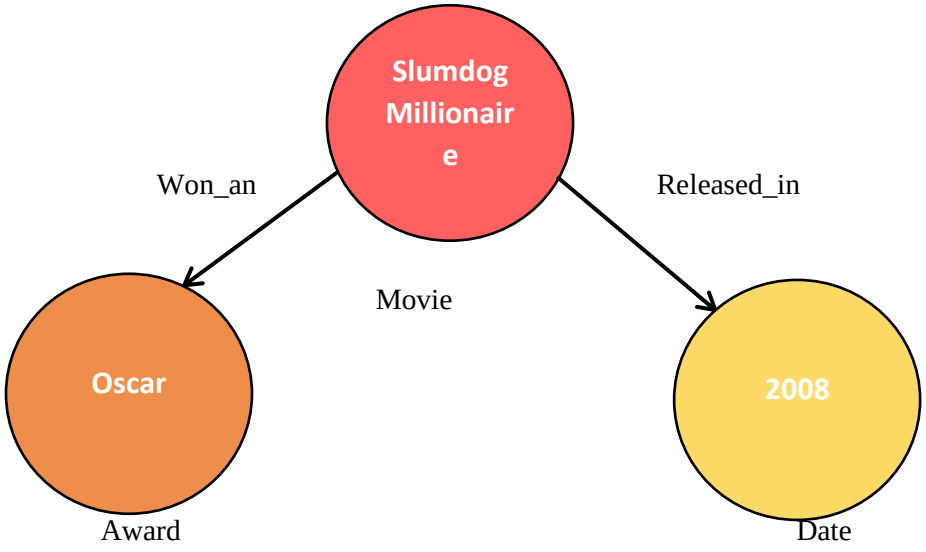

Figure 19: **Which movie released in 2008 won an Oscar?**

Chapter 5

Conclusion & Future Work

With the increasing interest in knowledge graphs-based question generation, there is a need of work that provides meaningful unique questions and estimates the difficulty level. We have done this research of automatic question generation from KG for Intelligent Tutoring Systems.

In this thesis we worked on creating a small KG for our experimentation and using it for question generation. A KG lacks meta information and does not recognize its entities as persons, places and etc. So before generating questions it was necessary to recognize the entities and for this purpose we used spaCy which is a natural language processing library, we used it with python and created a model of our recognized entities. After the model was created we used it for mapping for question generation. Mapping was done by using triples taken from KG 'Movies'. Our system generates questions of two difficulty levels i.e. easy and hard. Questions having easy difficulty level are generated by using a single triple, while more than one triples are used for the generation of questions with difficulty level as hard.

Our results showed that when more nodes/entities were involved in the generation of a question then the difficulty level was hard due to the increase in graph traversal complexity of BFS which is O(V+E), and similarly few nodes were involved in the generation of easy questions reducing the complexity. This shows that more complex query traversals are done for harder questions and simple cypher queries and graph traversals for easy level questions.

Future work could be done on using the 'concepts' present in KG for generation of questions with different difficulty levels as some concepts are tougher compared to others. Along with that the popularity of an entity could also be used for difficulty estimation as less popular entities as questions are harder to answer because people

wouldn’t know about that particular object, whereas questions related to more popular entities are easy to answer.

Furthermore, our system works with two triples for question generation and has two levels of difficulty i.e, easy and hard, for future work more than two triples could be involved in question generation and the levels of difficulty could be divided as easy, medium and hard.

Appendix

Easy Questions

s/no	Triples	Questions
1.	*('Keanu Reeves', 'born_in', 'Lebanon')*	*Where was Keanu Reeves born?*
2.	*('Keanu Reeves', 'born_on', 'September 2, 1964')*	*When was Keanu Reeves born?*
3.	*('Keanu Reeves', 'acted_in', 'The Matrix')*	*Who acted in The Matrix?*
4.	*('The Matrix', 'released_in', '1999')*	*When was The Matrix released?*
5.	*('Robert Zemeckis', 'directed', 'The Polar Express')*	*Who directed The Polar Express?*
6.	*(The Matrix Reloaded, second_sequel, The Matrix)*	*What was the second sequel to The Matrix?*
7.	*(The Matrix Revolutions, third_sequel, The Matrix)*	*What was the third sequel to The Matrix?*
8.	(*Jessica Thompson, reviewed, The Replacements)*	*Who reviewed The Replacements?*
9.	*('The Matrix', 'filmed_at', 'Sydney')*	*Where was The Matrix filmed?*
10.	*('The Matrix', 'earned', '$460 million')*	*How much The Matrix earned?*

11.	*('Joel Silver', 'produced', 'The Matrix')*	*Who produced The Matrix?*
12.	*('Lilly Wachowski', 'directed', 'The Matrix')*	*Who directed The Matrix?*
13.	*('The Polar Express', 'released_in', '2004')*	*When was The Polar Express released?*
14.	*('The Matrix Reloaded', 'released_in', '2003')*	*When was The Matrix Reloaded released?*
15.	*('Bride of Chuky', 'fourth_sequel', 'Childs play'),*	*What was the fourth sequel to Childs play?*
16.	*('Seed of Chuky', 'fifth_sequel', 'Childs play')*	*What was the fifth sequel to Childs play?*
17.	*('Jim Cash', 'wrote', 'Top Gun')*	*Who wrote Top Gun?*
18.	*('Top Gun', 'filmed_at', 'San Diego')*	*Where was Top Gun filmed?*
19.	*('Top Gun', 'costed', '$15 million')*	*How much Top Gun costed?*
20.	*('Top Gun', 'earned', '$815.5 million')*	*How much Top Gun earned?*

Difficult Questions

S/no	*Triples*	*Questions*
1.	*('Carrie-Anne Moss', 'acted_as', 'Trinity'),* *('Carrie-Anne Moss', 'acted_in', 'The Matrix')*	*Who acted as Trinity in The Matrix?*
2.	*(Keanu Reeves, acted_in, The Matrix)* *(Keanu Reeves, acted_as, Neo)*	*Who acted in The Matrix as Neo?*
3.	*(Slumdog Millionaire, released_in, 2008)* *(Slumdog Millionaire, won_an, Oscar)*	*Which movie released in 2008 won an Oscar?*
4.	*(Tom Cruise, born_on, July 3, 1962)* *(Tom Cruise, acted_in, A Few Good Men)*	*Who was born on July 3, 1962 acted in A few good men?*
5.	*(The Matrix Revolutions, third_sequel, The Matrix)* *(The Matrix Revolutions, released_in, 2003)*	*When was the third sequel to The Matrix released?*

6.	*(The Matrix Reloaded, second_sequel, The Matrix)* *(The Matrix Reloaded, filmed_at, Sydney)*	*Where was the second sequel to The Matrix filmed?*
7.	*(Joel Silver, directed, The Matrix)* *(Joel Silver, directed, Die Hard)*	*Who directed The Matrix and Die Hard?*
8.	*('V for Vendetta', released_in, 2006)* *('V for Vendetta', earned, $132.5 million)*	*Which movie released in 2006 earned $132.5 million?*
9.	*('Lilly Wachowski', directed, The Matrix)* *('Lilly Wachowski', produced, 'V for Vendetta')*	*Who directed The Matrix and produced V for Vendetta?*
10.	*('Tom Hank', 'acted_as, 'Mr. White')* *('Tom Hank', 'acted_in, 'That Thing You Do')*	*Who acted as Mr. White in That Thing you do?*
11.	*('The Artist', 'released_in', '2011')* *('The Artist', 'won_an', 'oscar')*	*Which movie released in 2011 won an Oscar?*
12.	*('Billy Crystal', 'born_in', 'New York City')* *('Billy Crystal', 'acted_in', 'When Harry Met Sally')*	*Who was born in New York City and acted in When Harry Met sally?*
13.	*('Nora Ephron', 'wrote', 'When Harry*	*Who wrote When Harry Met Sally*

	Met Sally') *('Nora Ephron', 'wrote', 'Sleepless in Seattle')*	*and Sleepless in Seattle?*
14.	*('Christina Ricci', 'born_on', 'February 12, 1980')* *('Christina Ricci', 'acted_as, 'Trixie')*	*Who was born on February 12, 1980 and acted as Trixie?*
15.	*('Bride of Chuky', 'fourth_sequel', 'Childs play'),* *('Bride of Chuky', 'released_in, '1998'),*	*When was fourth sequel to the Childs play released?*
16.	*('Seed of Chuky', 'fifth_sequel', 'Childs play')* *('Seed of Chuky', earned, $24.8 million)*	*How much fifth sequel of Childs play earned?*
17.	*('Bride of Chuky', 'fourth_sequel', 'Childs play'),* *('Bride of Chuky', costed', $25 million)*	*How much fourth sequel of Childs play costed?*
18.	*(Tom Cruise, born_on, July 3, 1962)* *(Tom Cruise, acted_as, 'Lt. Daniel Kaffee')*	*Who was born on July 3, 1962 acted as 'Lt. Daniel Kaffee?*
19.	*(Dev Patel, acted_in, Slumdog Millionaire)* *('Dev Patel', 'acted_as', 'Jamal Malik')*	*Who acted in Slumdog Millionaire as Jamal Malik?*

20.	*(Val Kilmer, acted_as, Iceman)* *(Val Kilmer, acted_in, Top Gun)*	*Who acted as Iceman in Top Gun?*

References

[1] Alqahtani, M., *An Overview: Intelligent Tutoring Systems*. 2011.

[2] D. Seyler, M. Yahya, and K. Berberich, "Knowledge questions from knowledge graphs," *arXiv preprint arXiv:1610.09935*, 2016.

[3] T. Alsubait et al, "Generating multiple choice questions from ontologies: Lessons learnt," *In OWLED*, 2014.

[4] H. Zafar, G Napolitano and J. Lehmann, "Formal Query Generation for Question Answering over Knowledge Bases," *In Extended Semantic Web Conference*, 2018.

[5] A. Abujabal, M. Yahya and M. Riedewald, "Automated Template Generation for Question Answering over Knowledge Graphs," *In WWW*, 2017.

[6] A. Abujabal, M. Planck, R. Saha and M. Yahya, "Never-Ending Learning for Open-Domain Question Answering over Knowledge Bases," *In WWW*, 2018.

[7] H. Li, C. Xiong and J. Callan, "Natural Language Supported Relation Matching for Question Answering with Knowledge Graphs," *In proceedings of the First Workshop on Knowledge Graphs and Semantics for Text Retrieval and Analysis (KG4IR)*, 2017.

[8] Ram, A., et al. *An Ontology of the Object Orientation for Intelligent Tutoring Systems*. in *2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT)*. 2017. IEEE.

[9] I. V. Serban, A. G. Duran, C. Gulcehre, S. Ahn, S. Chandar, A. Courville and Y. Bengio, "Generating Factoid Questions with Recurrent Neural Networks: The 30M Factoid Question-Answer Corpus," *In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

[10] Sharma, S., et al., *Survey of Intelligent Tutoring Systems: a review on the development of expert/intelligent tutoring systems, various teaching strategies and expert tutoring system design suggestions.* 2014. **3**(11): p. 37-42.

[11] R. S. Mittal, S. Nagar, M. Sharma, U. Dwivedi, P. Dey and R. Kokku, “Using a Common-Sense Knowledge Base to Auto Generate Multi-Dimensional Vocabulary Assessments,” 2018.

[12] Q. Guo, C. Kulkarni, A. Kittur, J. P. Bigham, and E. Brunskill, “Questimator: Generating Knowledge Assessments for Arbitrary Topics,” *In IJCAI,* 2016.

[13] S. Indurthi, D Raghu, M. M. Khapra and S. Joshi, “Generating Natural Language Question-Answer Pairs from a Knowledge Graph Using a RNN Based Question Generation Model,” *In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics,* 2017.

[14] D. Vrandeˇ ci´c, “Wikidata: A new platform for collaborative data collection,” *In Proceedings of the 21st International World Wide Web Conference, pages 1063–1064,* 2012.

[15] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, “Freebase: a collaboratively created graph database for structuring human knowledge,” *In Proceedings of the 2008 ACM SIGMOD Conference on Management of Data, pages 1247–1250,* 2008.

[16] K. Vanlehn, “The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems,” *Educational Psychologist, 46:4, 197-221,* 2011.

[17] D. Seyler, M. Yahya, and K. Berberich, “Generating quiz questions from knowledge graphs,” *In WWW,* 2015.

[18] R. Shah, D. Shah and L. Kurup, “Automatic question generation for intelligent tutoring systems,” *2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA),* Mumbai, 2017.

[19] E. V. Pérez, L. M. R. Santos, M. J. V. Pérez, J. P. de Castro Fernández and R. G. Martín, “Automatic classification of question difficulty level: Teachers' estimation vs. students' perception,” *2012 Frontiers in Education Conference Proceedings,* Seattle, WA, 2012.

[20] D. Hutzler, E. David, M. Avigal and R. Azoulay, “Learning Methods for Rating the Difficulty of Reading Comprehension Questions,” *2014 IEEE International Conference on Software Science, Technology and Engineering*, Ramat Gan, 2014.

[21] M. Yahya, K. Berberich, S. Elbassuoni, and G. Weikum, “Robust question answering over the web of linked data,” *In Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, pages 1107–1116. ACM*, 2013.

[22] J.H. Wolfe, “Automatic question generation from text-an aid to independent study,” *ACM SIGCUE Outlook* ***10****(SI)* ,1976.

[23] L. Zou, R. Huang, H. Wang, J. Xu Yu, W. He, and D. Zhao, “Natural language question answering over rdf: a graph data driven approach,” *In Proceedings of the 2014 ACM SIGMOD international conference on Management of data, pages 313–324. ACM,* 2014.

[24] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, “DBpedia: A Nucleus for a Web of Open Data,” *Asian Semantic Web Conference,* 2007.

[25] F. Suchanek, G. Kasneci, G. Weikum, “Yago: A Core of Semantic Knowledge Unifying WordNet and Wikipedia,” *International world wide web conference,* 2007.

[26] A. N. Ngomo et al, “Sorry, i don’t speak SPARQL: translating SPARQL queries into natural language.” *WWW* 2013.

[27] M. Kharrat, A. Jedidi and F. Gargouri, “SPARQL Query Generation based on RDF Graph,” *In Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management,* 2016.

[28] Yathongchai, W., et al., *SQL Learning Object Ontology for an Intelligent tutoring system.* 2013. **3**(2): p. 168.

[29] P. Pabitha, M. Mohana, S. Suganthi and B. Sivanandhini, "Automatic Question Generation system," *2014 International Conference on Recent Trends in Information Technology*, Chennai, 2014.

[30] Badaracco, M. and L. Martínez. An intelligent tutoring system architecture for competency-based learning. *in International Conference on Knowledge-Based and Intelligent Information and Engineering Systems. 2011. Springer.*

[31] Alkhatlan, A. and J.J.a.p.a. Kalita, Intelligent Tutoring Systems*: A Comprehensive Historical Survey with Recent Developments. 2018.*

[32] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. *URL http://www.w3.org/TR/rdf-sparql-query/.*

[33] George A. Miller. Wordnet: A Lexical Database for English. Communications of the ACM, *38(11):39–41, 1995.*

[34] N. Tandon, G. de Melo, F. M. Suchanek, and G. Weikum. WebChild: harvesting and organizing commonsense knowledge from the web. *In Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, February 24-28, 2014, pages 523–532. ACM, 2014*

[35] R. Navigli and S. P Ponzetto. 2012. BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. *Artificial Intelligence, 193, Elsevier, pp. 217-250.*

[36] "Freebase". *Google Plus*. 16 December 2014. Archived from the original on 20 March 2019.

[37] Markoff, John (2007-03-09). "Start-up Aims for Database to Automate Web Searching". *The New York Times*. Retrieved 2007-03-09.

[38] "So long and thanks for all the data!". 2 May 2016. Retrieved 5 May 2016.

[39] "Cypher Introduction". *Neo Technology. Retrieved January 31, 2017.*

[40] "Cypher: An Evolving Query Language for Property Graphs" (PDF). *Proceedings of the 2018 International Conference on Management of Data. ACM. Retrieved June 27, 2018.*

[41] M. Needham & Amy E. Hodler, Neo4j Graph Algorithms in Neo4j: Graph Algorithm Concepts

[42] Joy Chao, Community Graphista. Graph Databases for Beginners: Graph Search Algorithm Basics

[43] V. Rus, B. Wyse, P. Piwek, Mihai C. Lintean, S. Stoyanchev, and C. Moldovan. The First Question Generation Shared Task Evaluation Challenge. *In INLG 2010 - Proceedings of the Sixth International Natural Language Generation Conference, July 7-9, 2010, Trim, Co. Meath, Ireland. The Association for Computer Linguistics, 2010.*

[44] K. Sakaguchi, Y. Arase, and M. Komachi. Discriminative Approach to Fill-in-the-Blank Quiz Generation for Language Learners. *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers, pages 238–242. The Association for Computer Linguistics, 2013.*

[45] A. Narendra, M. Agarwal, and R. shah. Automatic Cloze Questions Generation. *In Recent Advances in Natural Language Processing, RANLP 2013, 9-11 September, 2013, Hissar, Bulgaria, pages 511–515. RANLP 2011 Organising Committee / ACL, 2013.*

[46] I. Labutov, S. Basu, and L. Vanderwende. Deep Questions without Deep Understanding. *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL*

2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers, pages 889–898. The Association for Computer Linguistics, 2015.

[47] Maha M. Al-Yahya. OntoQue: A Question Generation Engine for Educational Assesment Based on Domain Ontologies. *In ICALT 2011, 11th IEEE International Conference on Advanced Learning Technologies, Athens, Georgia, USA, 6-8 July 2011, pages 393–395. IEEE Computer Society, 2011.*

[48] J. Liu, Q. Wang, Chin-Yew Lin, and Hsiao-Wuen Hon. Question Difficulty Estimation in Community Question Answering Services. *In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, pages 85–90. ACL, 2013.*

[49] R. Herbrich, T. Minka, and T. Graepel. TrueSkill™ : A Bayesian Skill Rating System. *In Advances in Neural Information Processing Systems 19, pages 569–576. MIT Press, 2007.*

[50] J. Yang, L. A. Adamic, and M. S. Ackerman. Competing to Share Expertise: The Taskcn Knowledge Sharing Community. In ICWSM, 2008.

[51] K. Collins-Thompson and J. Callan. Predicting reading difficulty with statistical language models. *Journal of the American Society for Information Science and Technology, 56(13):1448–1462, November 2005.*

[52] M. Heilman, K. Collins-Thompson, and M. Eskenazi. An Analysis of Statistical Models and Features for Reading Difficulty Prediction. *In Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications, EANL'08,pages71–79,Stroudsburg,PA,USA,2008.AssociationforComputational Linguistics.*

[53] David Mack Answering English questions using knowledge graphs and sequence translation

[54] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, and C. Bizer, "DBpedia–a large-scale, multilingual knowledge base extracted from wikipedia," T*he Semantic Web, pages 167–195*, 2015.

[55] What is knowledge base? URL http://searchcrm.techtarget.com/definition/ knowledge-base

[56] Neo4j Overview URL https://www.tutorialspoint.com/neo4j/neo4j_overview.htm