Knowledge Base and Knowledge Graph

Jingyuan Zhang, 610220, jingyuanz@student.unimelb.edu.au
Jianxing Ma, 848424, jianxingm@student.unimelb.edu.au
Shukai Ma, 848482, shukaim@student.unimelb.edu.au

Abstract
Recently, the term Knowledge Base has been applied frequently both in academia and commercial, which is a large knowledge-based system of structured information about real-world entities and their relationships, along with the advent of linked data in semantic web. The popularity of Knowledge Base is obviously affected by the concept of Google's Knowledge Graph in 2012 for improving its searching services. There are also some open available knowledge graphs like YAGO, DBpedia which are extracted from semi-structured knowledge or collected from the web. This article will provide an introduction to Knowledge graph and knowledge base and its existing products and primary applications. It will also demonstrate how to construct a knowledge graph by presenting the key technique of Information Extraction. Finally, the article will present approaches and algorithms for improving the utility of these knowledge graphs and discussion of related future work.

TABLE OF CONTENT

1. Introduction

1.1 Motivation and Introduction to Knowledge Graph

As the concept of the semantic web is proposed, a growing number of Linked Open Data and User Generated Contents are posted on the Internet, the Internet has evolved from a document World Wide Web that contains hyperlinks between web pages to a Data World Wide Web that describes rich relationships between entities and entities (Demartini, 2016). In this context, Google in 2012 introduced Knowledge Graph as a semantic enhancement of Google's searching service which enables searching for various entities and concepts in the real world as well as relationships between entities rather than matching strings (Wöß and Ehrlinger, 2016).

1.2 Applications & Brief discussion of major approaches

There are various methods of constructing knowledge graphs. They can be edited like Freebase and Wikidata and can be built by extracting from large-scale semi-structured web knowledge bases like Wikipedia such as DBpedia and YAGO, by information extraction from unstructured information like NELL, PROSPERA and KnowledgeVault (Paulheim, 2016). Those applications differ in characteristics, architecture, operational purpose and techniques used.

1.3 Brief overview of the remaining body of the report

The rest of this article is organized as follows. Section 2 presents an introduction to Information Extraction along with its architecture and main approaches, which is a key component in Knowledge Graph. In section 3 and section 4, it is illustrated how to construct a knowledge graph using Neo4j and several existing knowledge base products. Section 5 discusses knowledge graph inference including main algorithms, knowledge graph fusion and correction, and future work. Applications of Knowledge graph and Knowledge base are stated in section 6.

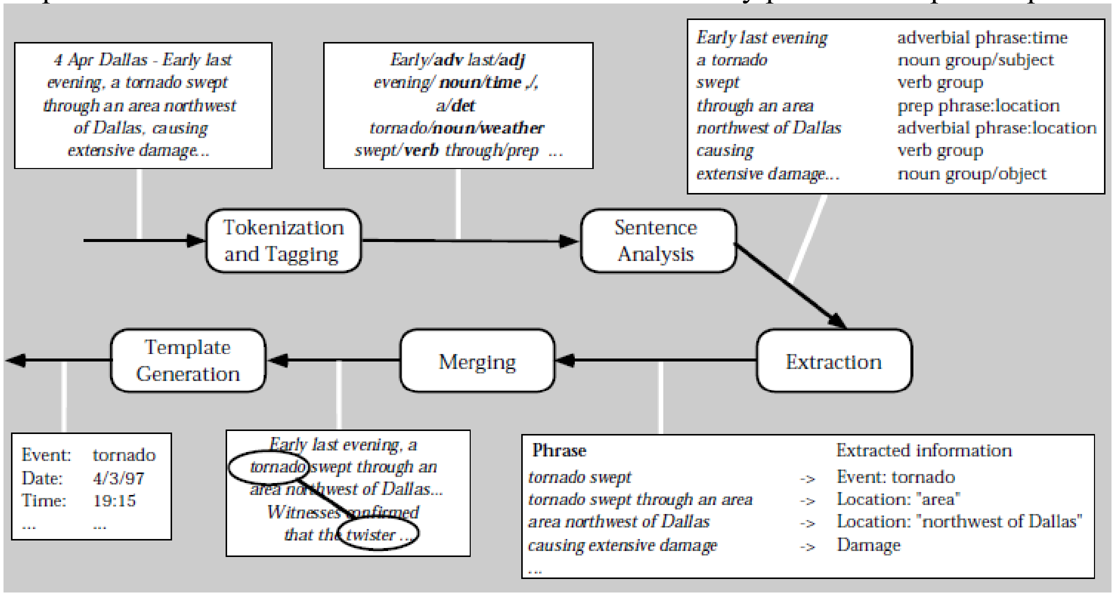
2. Information Extraction for Knowledge Graph

2.1 Introduction to Information Extraction and Knowledge Acquisition

Information Extraction (IE) is initially proposed by the DARPA MUC program (Message Understanding Conference) in 1987. MUC originally defined IE as tasks of extracting specific, well-defined information from text of homogeneous documents in specific domains and filling pre-defined templated or form slots with extracted information (Nedellec, Nazarenko and Bossy, 2016). With roots in Knowledge Graph and Knowledge Base, IE refers to a process where information such as entities, relationships between entities and attributes describing entities is extracted from unstructured text and turned into structured data to construct Knowledge Graphs (Sarawagi, 2007). This allows much wealthier forms of searching on various unstructured sources rather than possible searching with only keyword. IE is also beneficial to several text and web applications such as integration of product information from various websites, removal of noisy data, finding the proteins illustrated in a biomedical journal and question answering systems (Tang, et al, 2008).

2.1.1 Information Extraction Systems Architecture

Information extraction systems built for diverse tasks often vary from each other. Previous literatures about Information Extraction Systems demonstrated several different architectures for it. Based on traditional NLP systems, Cardie (1997) proposed a five-step system shown in figure 2.1-1. Where each input text is initially separated into sentences or words in tokenization and tagging phase followed the sentence-analysis phase. The extraction phase is regarded as domain-specific component of the system, in which domain-specific relations among entities are identified. The merging phase focus on co-reference resolution or anaphora resolution. The template-generation phase determines the number of distinct events, maps the extracted information onto each event and finally produces output templates.



4 Apr Dallas - Early last evening, a tornado swept through an area northwest of Dallas, causing extensive damage...
Early/adv last/adj evening/ noun/time ,/, a/det tornado/noun/weather swept/verb through/prep ...
Early last evening — adverbial phrase:time
a tornado — noun group/subject
swept — verb group
through an area — prep phrase:location
northwest of Dallas — adverbial phrase:location
causing — verb group
extensive damage... — noun group/object
Tokenization and Tagging
Sentence Analysis
Extraction
Merging
Template Generation
Event: tornado
Date: 4/3/97
Time: 19:15
...
Early last evening, a tornado swept through an area northwest of Dallas... Witnesses confirmed that the twister ...
Phrase
Extracted information
tornado swept -> Event: tornado
tornado swept through an area -> Location: "area"
area northwest of Dallas -> Location: "northwest of Dallas"
causing extensive damage -> Damage
...

Figure 2.1-1

A modern simple version of IE architecture with focus on Named Entity Recognition and Relation Extraction is proposed by Bird, Klein and Loper in 2015 shown in figure 2.1-2. It begins by processing a raw text using several procedures which including sentence segmentation, tokenization, POS tagging that will be discussed later and prove very helpful for the next step, NER. Finally, Relation Extraction is applied to search for likely relations between different entities.

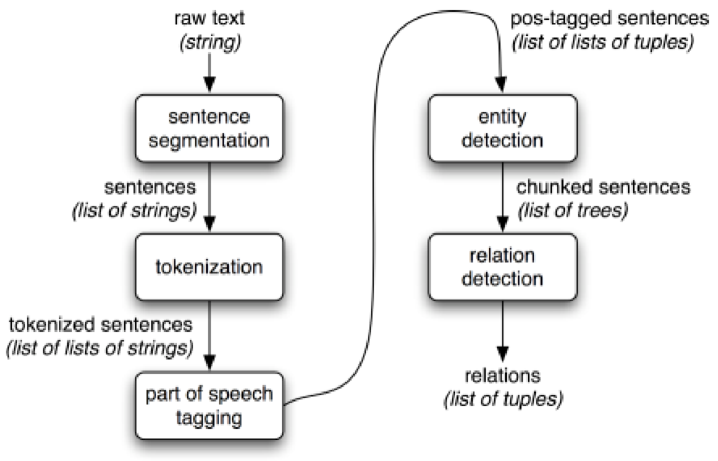


Figure 2.1-2

2.1.2 IE Tasks with examples

The task of IE aims to recognize instances of a specific pre-defined class of entities, relationships and events in natural language texts, as well as extraction of the related attributes of identified entities, relation or events. Classic IE tasks mainly include Named Entity Recognition, Co-reference Resolution, Relation Extraction and Event Extraction (Piskorski and Yangarber, 2012). Named Entity Recognition (NER) refers to detect each mention of a named entity in text and classify its types (Jurafsky and Martin, 2016). For example, organizations (e.g. World Health Organization), place names (e.g. the Great Wall), temporal expression (e.g. 1 May 2018) currency expressions (e.g. 20 Million U.S dollars) and so on. Next, co-reference resolution is to identify or link multiple mentions of the same entity behind these mentions in the text. For instance, to identify the mentions of UK and Britain refer to the same real-life entity. Relation Extraction (RE) requires to detect and categorize semantic relations among the identified entities in the text, which is often binary relations like child-of, part-whole. Finally, event Extraction (EE) is to find events where the identified entities participate and produce structured detailed information about them. Typically, EE refers to extraction of several entities and relations among them.

2.2 Approaches to IE

2.2.1 Early years: Knowledge Engineering (KE) Approaches

The Knowledge Engineering approaches detect and extract the target information from text according to linguistic knowledge in the form of rules or patterns that created by human experts via inspection of the test corpus and intuition, or rather, persons who are professional in IE system and formalism for communicating patterns for the system, write rules for IE process used to extract information. The FASTUS system is an instance of a robust general-purpose IE system for processing English and Japanese, where each stage in the process is denoted by a finite-state device and output structures of the sequential stages act as input for the following stage.

Building a high performance KE-based system is typically an iterative process, where a series of rules is defined, the system is run over a training corpus, and the output is used to check whether the rules can be improved. Namely, the knowledge engineers define rules, make appropriate adjustments on the rules, and iterates this process.

2.2.2 Automatically Trainable IE Systems

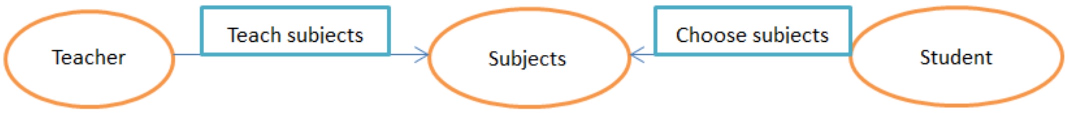
Trainable IE systems are motivated using machine-learning techniques to unburden workload on customizing general-purpose IE systems, which move human effort from knowledge engineering toward annotating training data for input of machine-learning algorithms. There are three most popular methods for automatic IE, including sequential labeling based extraction methods, classification based extraction methods and rule based methods (Tang, et al, 2008).

2.2.3 Comparison of approaches and discussion in different situations

In KE-based approach, grammars are constructed by hand, domain patterns are discovered by human specialists and much labor is required. Another shortcoming is modifications on rules can be hard. However, it is preferred when resources like lexicons or rule writers are available, training data is expensive to obtain or extraction specifications are likely to be changed (Appelt and Israel, 1999). From the computational view, finite-state devices are efficient according to the closure attributes and the existing effective transformation and optimization algorithms. From linguistic viewpoint, the most of linguistic phenomena can be easily shown by finite-state devices and broken into several autonomous finite-state sub-descriptions.
Compared with KE-based approaches, trainable IE systems focus on learning rules from annotated corpus and user interactions and training input data using statistical methods for machine-learning algorithms. For a scientist, this system is more attractive when there is no skilled rule writer, extraction requirements are stable and good performance is required (Appelt and Israel, 1999).

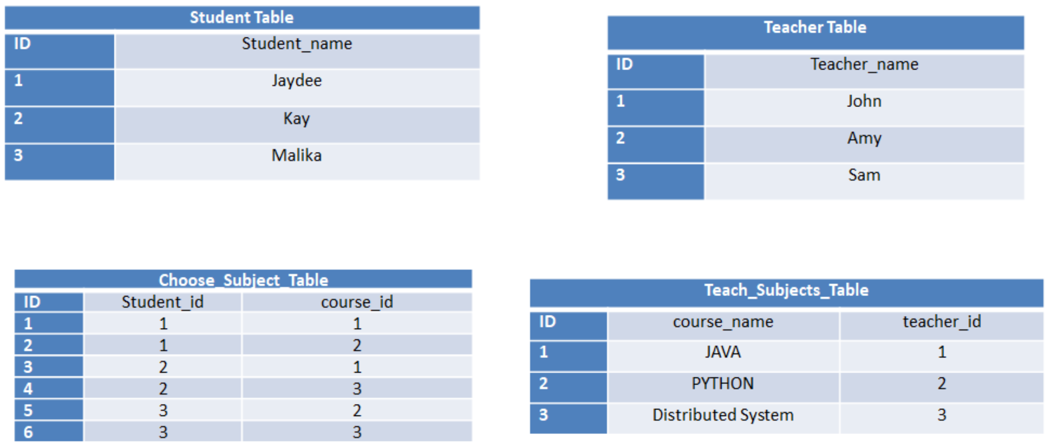
3. Construct a Knowledge Graph using Neo4j

Neo4j is a graph database management system using cypher command to construct graph database. Cypher is a declarative, SQL-inspired language which can describe patterns in graphs by means of asci-art syntax. Compared with other database language such as MYSQL, Neo4j using cypher can construct a multi-relational database more efficiently. In addition, refer to the triple relation or RDF used in Knowledge base in the last section cypher is also an enhanced choice. Assuming there is a subject selection development project needs to be delivered. In this project, there are two main triple relations, which are teacher teaches the subject and student choose the subject. To present such relations in this project, the RDF model for this project is shown in the graph 3-1 below.



Graph 3-1

Using database languages such as MYSQL in developing such a project would be complicated. If the MYSQL is used in the construction of the databases, at least those four tables in graph 3-2 should be created. From those tables in graph 3-2, it can be noticed that, in order to obey the Third Normalization form, each tuple in the table is assigned an ID, which needs extra storage space. In addition, so as to develop such an N-N-relation system, the complicated relational algebra as well as other annoyed issues such as the keys for projections are needed in this project. To conclude, using database language such as MYSQL is not suitable in developing such a multi-relational database.



Student Table

ID	Student_name
1	Jaydee
2	Kay
3	Malika

Teacher Table

ID	Teacher_name
1	John
2	Amy
3	Sam

Choose_Subject_Table

ID	Student_id	course_id
1	1	1
2	1	2
3	2	1
4	2	3
5	3	2
6	3	3

Teach_Subjects_Table

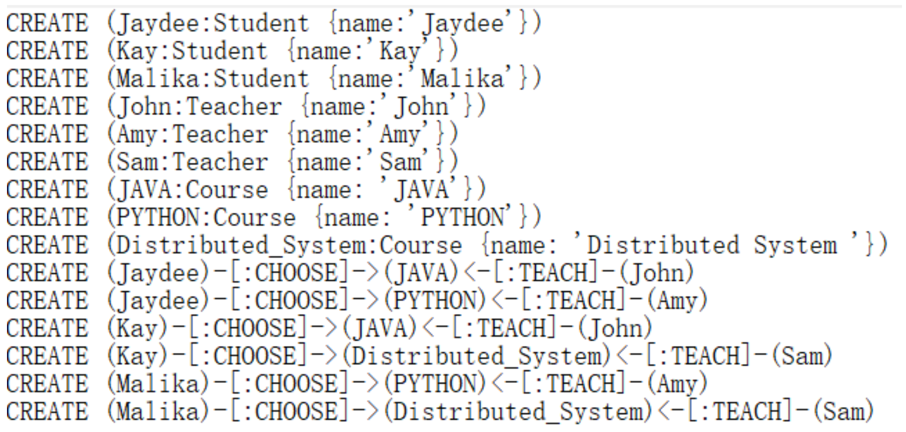
ID	course_name	teacher_id
1	JAVA	1
2	PYTHON	2
3	Distributed System	3

Graph 3-2

If the Neo4j or cypher is used in this project, it would be much straightforward. In cypher the RDF model shown in graph 3-1 can be abstracted as the cypher command shown in graph 3-3 below. In this situation, no more tables are needed to be created; the process to input the data is actually the process to create the database. In this project, it is convenient to use the cypher commands shown in graph 3-4 below to input the entities and relations. Furthermore, after finishing the process of input, the graph database is constructed in Neo4j, which is shown as in graph 3-5 below. Compared with the last example using MYSQL, this approach is much more straightforward and efficient.

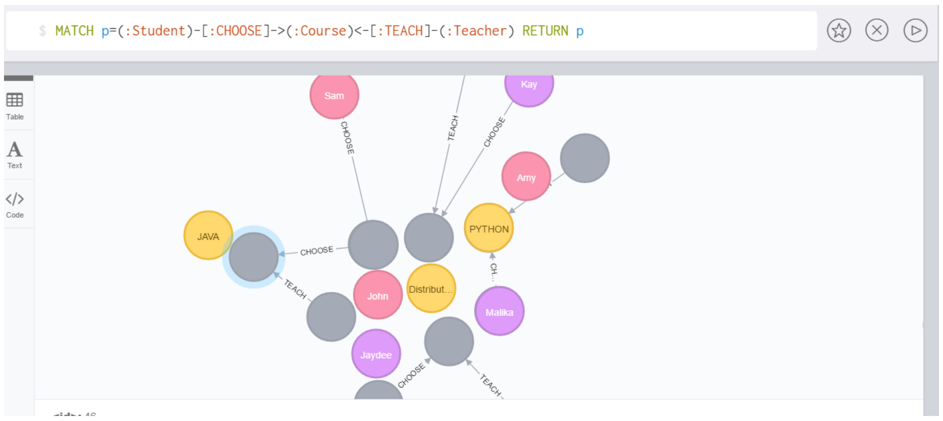
```
(:Student{name})-[:CHOOSE]->(:Course {name})<-[:TEACH]-(:Teacher{name})
```

Graph 3-3



```
CREATE (Jaydee:Student {name:’Jaydee’})
CREATE (Kay:Student {name:’Kay’})
CREATE (Malika:Student {name:’Malika’})
CREATE (John:Teacher {name:’John’})
CREATE (Amy:Teacher {name:’Amy’})
CREATE (Sam:Teacher {name:’Sam’})
CREATE (JAVA:Course {name: ’JAVA’})
CREATE (PYTHON:Course {name: ’PYTHON’})
CREATE (Distributed_System:Course {name: ’Distributed System ’})
CREATE (Jaydee)-[:CHOOSE]->(JAVA)<-[:TEACH]-(John)
CREATE (Jaydee)-[:CHOOSE]->(PYTHON)<-[:TEACH]-(Amy)
CREATE (Kay)-[:CHOOSE]->(JAVA)<-[:TEACH]-(John)
CREATE (Kay)-[:CHOOSE]->(Distributed_System)<-[:TEACH]-(Sam)
CREATE (Malika)-[:CHOOSE]->(PYTHON)<-[:TEACH]-(Amy)
CREATE (Malika)-[:CHOOSE]->(Distributed_System)<-[:TEACH]-(Sam)
```

Graph 3-4



```
$ MATCH p=(:Student)-[:CHOOSE]->(:Course)<-[:TEACH]-(:Teacher) RETURN p
```

Graph 3-5

Having known how to create a graph database using Neo4j, it is the time to construct the knowledge. To construct a knowledge base, the preprocessed entities and relations are necessary. Today, one possible approach to construct a knowledge base could be that simply extract the entities, relations or graphs from other open-source knowledge bases. That approach is actually what the most companies choose to construct their own knowledge bases: previous approaches for building knowledge bases primarily rely on the existing knowledge repositories. However, this approach would result in a large amount of noises due to the diverse RDF triples in different knowledge bases. For example, there would be a lot of alias exist when extract entities or relations from two or more different open-source knowledge bases. Alias contains synonyms, abbreviations, international spelling variations, etc. For instance, color and colour is a pair of alias, and Obama and “/person/Obama” is another example of alias. In the RDF model used in knowledge graph, nodes, which are refer to entities, are connected by edges, which are relations. Therefore, one possible solution could be that: adding a new edge to connect two aliases, which is called alias edge. Refer to the last example, alias edge could be as: “color – alias - colour”. After adding the alias edges, the nodes which are a pair of alias would be able to share their relations, which would significantly enhance the dimensions of the knowledge base.

4. Current Knowledge Base Products

In recent years, an explosion of large-scale knowledge bases has been launched, including academic project such as YAGO and DBpedia, as well as commercial projects such as Google KG and others. Refer to the last section, an increasing number of works today are aiming at automatic knowledge base construction; to be more specific, the approaches to automatic knowledge base construction can be grouped into 4 main types: (1) knowledge bases that are constructed based on Wikipedia infoboxes and other structured data sources; (2) approaches choose schema-less extraction techniques applied to the whole web; (3) knowledge bases extract data and facts from the whole web but select a static model (for instance, a probabilistic model) to filter data, such as DeepDive, and (4) knowledge base constructs taxonomies (is-a hierarchies). Table 4-1 below shows some typical knowledge bases and the comparisons among them in data source, language, number of entities and number of relations. Based on this table, it can be noticed that these knowledge bases reposit massive facts about the world, such as information about individuals, place as well as dates, which are generally mentioned as entities.

ID	Knowledge Bases	Data Source	Language	# of Entities	# of Relations
1	WordNet	Multi	English	25 Million	600 Million
2	Freebase	Wikipedia	English	16 Million	270 Million
3	YAGO	Wikipedia	Multi	10 Million	120 Million
4	DBpedia	Wikipedia	Multi	28 Million	9.5 Billion
5	Open IE	Text-based	English	10 Million	120 Million
6	Google KG	Multi	Multi	20 Million	300 Million
7	CN-DBpedia	Wikipedia	Chinese	17 Million	200 Million
8	Probase	Text-based	English	2.7 Million	200 Million
9	Reverb	Multi	English	11 Million	400 Million
10	DeepDive	Multi	Englsih	20 Million	1 Billion

Table 4-1

This section will introduce the most popular knowledge base products: YAGO, DBpedia and DeepDive in a deeper degree. YAGO (Yet Another Great Ontology) is an open source knowledge base introduced by the Max Plank Institute. It was firstly released in 2008, and as of 2012, YAGO3 was updated to store over 10 million entities and contain over 120 million facts around these entities. The data sources which YAGO extracts from are mainly from Wikipedia for its categories and redirects, WordNet for its synsets and hyponymy and GeoNames for its entities names (MPN.com, 2015). The accuracy of YAGO was manually assessed to be more than 95% on a sample of facts, which seems to be reassuring. YAGO3 is provided in Turtle and TSV formats, and the source code of YAGO3 is available on GitHub. DBpedia is a crowd-sourced community work to abstract structured data from the information offered in diverse Wikimedia projects. It was developed by Leipzig University, University of Mannheim and OpenLink Software, and was firstly released in 2007. In October 2016, the stable version was released and available on GitHub. The structured information of DBpedia is an open knowledge graph which is accessible for anyone on the web. Actually, DBpedia data is served as Linked Data, which is revolutionizing the way applications interact with the Web. DBpedia describes approximately 28 million of entities and 9.5 billion of relations. Therefore, such a rich source of structured cross-domain knowledge is fertile ground for the development of AI systems (DBpedia.org, 2018). Lastly, DeepDive, as one of the approaches extracts data from the whole web but use a fixed schema, is a relatively advanced sort of data analysis system that offers a complete framework of extract knowledge graph relations from raw text. Compared with traditional systems, DeepDive differs in several ways. Firstly, DeepDive recognizes that data is often noisy and imprecise, so DeepDive calculates adjusted probabilities for each sentence. For instance, if DeepDive yields a fact with probability 0.95, the fact is 95% probably to be true. In addition, as it is stated before, DeepDive can utilize massive data from a variety of sources: applications constructed using DeepDive have extracted information from millions of documents, web pages, tables, and figures. Furthermore, DeepDive also can be regarded as a scalable, high-performance inference and learning engine (Palomares, 2016).

5. Knowledge Graph Inference

With the growing maturity of a diverse of Entity Extraction (EE) and Relation Extraction (RE) algorithms, countless triples of entities with predicates can be obtained from various types of data sources. Many existing Knowledge Base management tools help with automatically build a structured knowledge graph from those collected raw triples, enabling big corporations or academic organizations to build knowledge bases containing up to billions of nodes and edges using such pipeline. Furthermore, integrations (or merges) of large scale knowledge bases have also been realized recently, e.g. Google Knowledge Vault. However, even with the most state-of-the-art knowledge base, it is impossible to have it cover all entities and relations that are potentially mentioned in all possible queries. For example, we might see some data stating that Kay and Tom and students in The University of Melbourne, and another document saying that Jerry enrolled in COMP90001 and COMP90001 is a subject in unimelb. A query asks who are the students in unimelb. In this case, when a knowledge base is completed, we are very likely to see both Kay and Tom have a "StudentIn" relation with Unimelb, while Jerry does not have a direct connection with unimelb. From the perspective of human beings, we clearly know that Jerry is also a student in Unimelb by inferring with combinational facts, but it is hard for computers to find the same answer with a missing direct relation. There exist tremendous missing or paraphrased links in a knowledge graph similar to the example we show. In order to respond to a query that involves such links, it is urgent that we find out an approach to make the knowledge graph intelligent enough to reason over existing facts for new knowledge. This task, or area of research, is called Knowledge Graph Inference, also known as Knowledge Graph Completion. The report will focus on introducing three of the existing methods aiming to solve the tasks of link prediction and knowledge graph completion: A classic algorithm proposed by Lao and Cohen (2010b) called Path Ranking Algorithm (PRA), Knowledge Base Embedding models and Subgraph Feature Extraction algorithm.

5.1 Knowledge Graph Inference algorithms

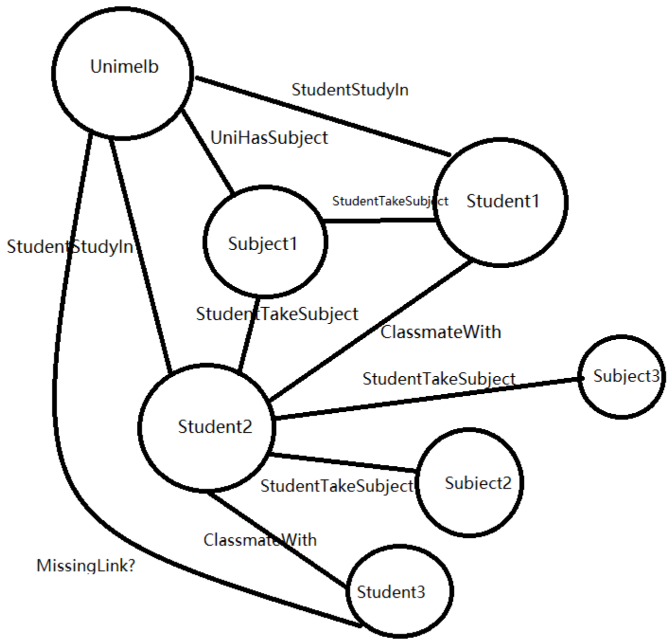
5.1.1 Path Ranking Algorithm

PRA algorithm is a graph based algorithm that treats the Knowledge Base as a graph composed of nodes and edges. The algorithm assumes that for any pair of entities having a specific relation, the subgraph structure around and in between these two entities should be similar to each other. Based on this assumption, the PRA algorithm computes a feature vector for each pair of entities as a sample instance used to train a logistic classifier for a particular relation type. Each feature vector consists of columns which represent possible path types between the source entity and the target entity. PRA assumes that if any pair of entities have a similar routing structure as existing triples, it is highly likely to also satisfy the predicate in that triple. As an example, if a classifier is trained to predict a missing “student study in” link (figure 1) between Unimelb and Student3, it has to learn from existing subgraph structures of entity pairs that are already linked with “student study in” relation, in this case, (Student2,Unimelb) and (Student1,Unimelb). Possible alternative routes between the two positive instances include “student takes subject” + “uni has subject” or “classmate with” + “student study in university” etc.. These paths used as features can be found using graph search algorithms like breadth first search and depth first search. In knowledge graphs that contain billions of entities, the amount of edges between nodes can become surprisingly large, thus overly inflate the dimension of feature vectors. A good way to easily control feature size is to simply set a threshold for search depth.
After deciding on features to be used for each instance, we compute a feature matrix similar to the example in Table 5.1.1-1 where each row is an instance for training data.

Feature Table	ClassmateWith+StudentStudyIn	StudentTakeSubject+UniHasSubject
(Student1, Unimelb)		
(Student2, Unimelb)		

Table 5.1.1-1

Yet we still need to fill in the values for those cells. PRA performs random walks in the graph from the source entity to the target entity, and calculates probabilities of reaching the target via various featured paths. This procedure is very similar to the simulation step in Monte Carlo Tree Search (Gilks et al., 1995). For example, in figure 1, if we follow the "StudentTakeSubject+UniHasSubject" route from student2, we end up reaching Unimelb for a chance of 0.5 because taking the "StudentTakeSubject" path can sink us to Subject2 entity, which is not linked with Unimelb. After filling into values of the feature matrix for all chosen paired instances, PRA will then be able to train any effective discriminative model (Logistic Regression, as proposed in the original paper) to predict new pairs with the probability produced by the classifier that the two entities are linked by a particular relation. To balance the training data, usually we also introduce negative sampling techniques for computing the feature table. We sample as many as negative samples by selecting a random entity to replace one of the source or the target entity for each positive instance. Such operation will dramatically increase the model performance.



However, this algorithm has some apparent drawbacks. Since rarely does a pair instance have all types of featured links, leading to an overwhelmingly feature sparsity. The sparsity may also suffer from the fact that PRA is incapable of distinguish between paraphrased links or links share similar meanings, hence creating duplicate feature types in its feature vector. The sparsity will negatively impact on model performance and generalisability. There are multiple feature selection approaches to cut the feature size, including the depth-limited breadth first search mentioned above and other statistical methods (e.g. information gain), but the sparsity does not necessarily get improved. Another disadvantage is the high computation complexity of this algorithm. The complexity breadth first search and random walks all heavily depend on the average out degrees of nodes in the knowledge graph (or connectivity of the graph). For a huge knowledge graph like Google Vault or Freebase, the computation complexity is non-trivial if sufficient amount of random walks are required to guarantee reliability of the probabilistic values.

5.1.2 Vector Space Model (VSM)

Another direction of research in Knowledge Graph Inference devotes their efforts to learning a vector space representation for entities and relations in the Knowledge Graph. The vector space model is also called Knowledge Graph Embedding method. By embedding the huge knowledge graph a low dimensional space, we can easily represent entities with fixed length vectors and a link between entities with their vector offsets. For example, when successfully finding a knowledge graph embedding vector space, it is likely that the vectors of "Obama + leaderOf ≈ UnitedStates" holds. If there is a missing link between, say, Xi Jinping and China, it is expected that adding the same vector as "leaderOf" to Xi will lead to a smoothest translation between two entities and sum up to a vector that is closest to China. Apart from predicting new links, this method also helps with comparisons or disambiguations between similar relations or entities via vector cosine similarity. For example, the vector of "studyIn" is expected to be similar to a vector of "isStudentOf" in the learned vector space. Among diverse models designed following this idea, we introduce a canonical machine learning based model called TransE (Bordes et al,. 2013) as a representative.
TransE learns the correct entity and relation representations by first randomly initializing a fixed length vector for all entities and relations. In learning phase, the model samples from huge training dataset a batch of valid(h, l, t) triples where h and t are entities and l is the link in between, while a negative batch is also sample consisting of false (h', l, t') where no link of l connects h' and t'. The model optimizes by minimizing a margin loss whose formula is given in figure 5.1.2-1 where γ is the margin and d is a distance (or dissimilarity) measurement function. In other word, the model tries to reward correct relational transfers and penalize incorrect ones. The model then updates the embeddings using batch gradient descent or other optimizers on the calculated loss.

$$\mathcal{L} = \sum_{(h,\ell,t)\in S} \sum_{(h',\ell,t')\in S'_{(h,\ell,t)}} \left[\gamma + d(\boldsymbol{h}+\boldsymbol{\ell},\boldsymbol{t}) - d(\boldsymbol{h'}+\boldsymbol{\ell},\boldsymbol{t'})\right]_+$$

Figure 5.1.2-1

Similar models include TransH (Wang et al,. 2014), TransR (Lin et al.,2015) and RESCAL (Nickel et al., 2011) all aiming at similar vector space transformation but with different architectures. Among them, TransR model is proposed to further solve a problem that TransH and TransE are only able to represent relations between every two entities with a single relation vector, while in natural cases, entities have links of various aspects. For example, if Kay and Tom are twins that study in the same college, they may be connected by both “BrotherOf” and “ClassmateWith” relations, hence inadequate to be represented by a single vector offset between vectors of Kay and Tom. TransR learns a separate vector space for any relation r as well as a single space for all entities. Entities will be transformed to the particular vector space owned by a relation before being assessed for existence of the relation, which consequently allows multiple relations between entities. Figure 5.1.2-2 shows a clear and simple illustration for this idea.

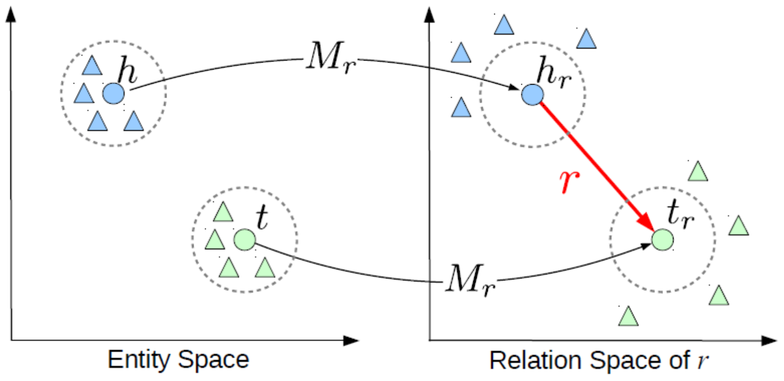


Figure 5.1.2-2

5.1.3 Subgraph Feature Extraction

In 2015, a new approach called Subgraph Feature Extraction (Gardner et al,. 2015) (SFE hereafter) was experimented to have surpassed Path Ranking Algorithm and overcome its demerits using a similar PRA style feature matrix. Unlike PRA, SFE eliminates the highly computationally expensive random walk simulation step, replacing the probabilities with binary features (1 for path existence and 0 for non-existence). Table 5.1.3-1 shows that the random walks are truly just a waste of computation, with both lower speed and precisions comparing to binary features (p.s. Not related to the report but wonderring why the author did not naturally experiment with binaries in the first place).

Dataset	**Method**	**MAP**
Freebase	Probabilities	.337
	Binarized	.344
NELL	Probabilities	.303
	Binarized	.319

Table 5.1.3-1

The saved computation resources are then used to extend the feature vectors by introducing additional features. New features include One-sided features, Vector Space features and Any-relation features, which are briefly explained next.

One-sided features are paths that do not necessarily connect two entities, but rather

end at one side, representing some properties of the entities, e.g. “gender:male” or “age:40”. These features imply important information that are useful to predicting relations. For example, knowing the gender and ages of entities may help distinguish between “SonOf”, “DaughterOf” and “MarriedTo”.
Vector Space Similarity features make good use of the vector space model. For every featured path, SFE replaces one edge at a time with an edge that is with a high vector similarity score. For example, an original PRA-style feature “ClassmateWith”+”StudyIn” might be morphing into other features like “ClassmateWith”+”IsStudentOf” or “RegisterInSameSubjectWith”+”StudyIn”. This will enrich the candidate features to better capture paraphrased relations, especially for knowledge graphs that is built heavily upon natural text data sources, where vast majority of relations do not have a formally defined type.
Any-Relation features take full advantages from the vector space features by further replace edges with another special edge type called “AnyRelation” apart from similar edges retrieved from vector space. The “AnyRelation” edge is simply a placeholder that can represent any type of edge. For example, the “ClassmateWith”+”IsStudentOf” feature may be further transformed into “AnyRelation”+”IsStudentOf” and “ClassmateWith”+”AnyRelation”.
There are also some other type of additional features that contribute to the finalized model, including bi-gram features, comparison features and etc.. Table 5.1.3-2 shows the experiment results for concatenating extra features.

Feature Types	MAP	MRR	Features
PRA-style features	.431	.806	240k
+ Comparisons	.405	.833	558k
+ One-sided	.389	.800	1,227k
+ One-sided + Comps.	.387	.817	1,544k
+ Bigrams	.483	1.00	320k
+ Vector similarity	.514	.910	3,993k
+ Bigrams + vec sim.	.490	.950	4,073k
+ Any Rel	**.528**	.933	649k

Table 5.1.3-2

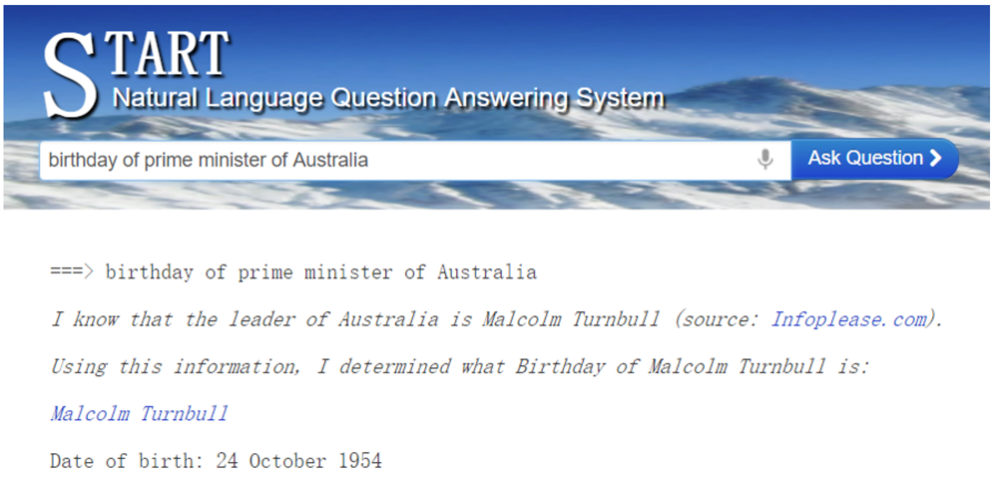
5.2 Knowledge Graph Fusion and Correction

The knowledge graph inference techniques are not only useful when predicting missing relations in the graph, but they can also play an essential role in improving the reliability of existing triples. Recently, Google has demonstrated their upgraded knowledge graph, renaming it The Knowledge Vault (Dong et al., 2014). The paper proposes that they have created the most all-round, large-scale and trustworthy structured knowledge base in history by merging information from existing open source knowledge bases as well as extracting from all kinds of data resources. They clarify that making use of trained logistic classifiers belonging to their knowledge base inference system, they can filter out noisy and unreliable links in existing knowledge graph. This can be summarized as simply as, for a specific entity pair, we assess a particular relation by computing the required features for the classifier on that relation, producing a probability from the model, and remove that link between entities if the probability is below some predefined threshold.

6. Applications of Knowledge graph and Base

6.1 Question Answering System

Question Answering ("QA") is the task of automatically determining the answer for a natural language question. Its primary approach is automatically construct a query and answer question relative to fixed structural databases, which is known as Natural Language Interface to Database or NLIB. The first phase of QA research focused on NL interface to structural database from a particular domain with idiosyncratic data semantics. This is because in the mid of 20 centuries, the scalability of the structural database was inherently limited in scale by the size. Actually, the development of the QA mainly focuses on two aspects: one is the process on the query including the algorithms of semantic parsing, etc., and the other is the size of the structural database where the answer is derived from. Based on this fact, the appearance of knowledge base significantly improves the second aspect of QA system. Refer to the last section, knowledge base stores millions of facts and relations about the world; thus, the approach now would be change to "automatically construct a query and answer question relative to knowledge base". Compared with the traditional structural database, knowledge base stores nearly as thousand times of data as traditional structural database. Therefore, using knowledge base in the QA system will significantly improve the efficiency, accuracy and abundancy of the QA system. Using Start, which is the world's first Web-based question answering system, as an example. The graph 6-1 below shows how it works when you want to know "the birthday of prime minister of Australia" in Start.


START
Natural Language Question Answering System
birthday of prime minister of Australia
Ask Question

===> birthday of prime minister of Australia

I know that the leader of Australia is Malcolm Turnbull (source: Infoplease.com).

Using this information, I determined what Birthday of Malcolm Turnbull is:

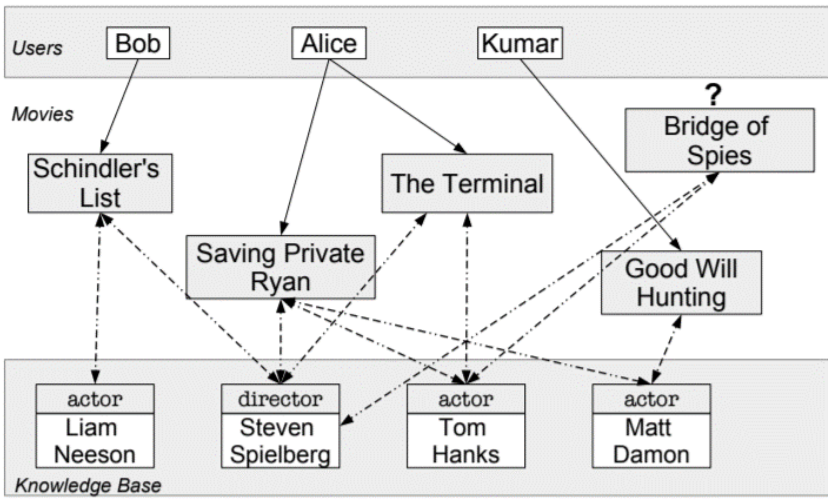
Malcolm Turnbull

Date of birth: 24 October 1954

Graph 6-1

6.2 Recommendation System

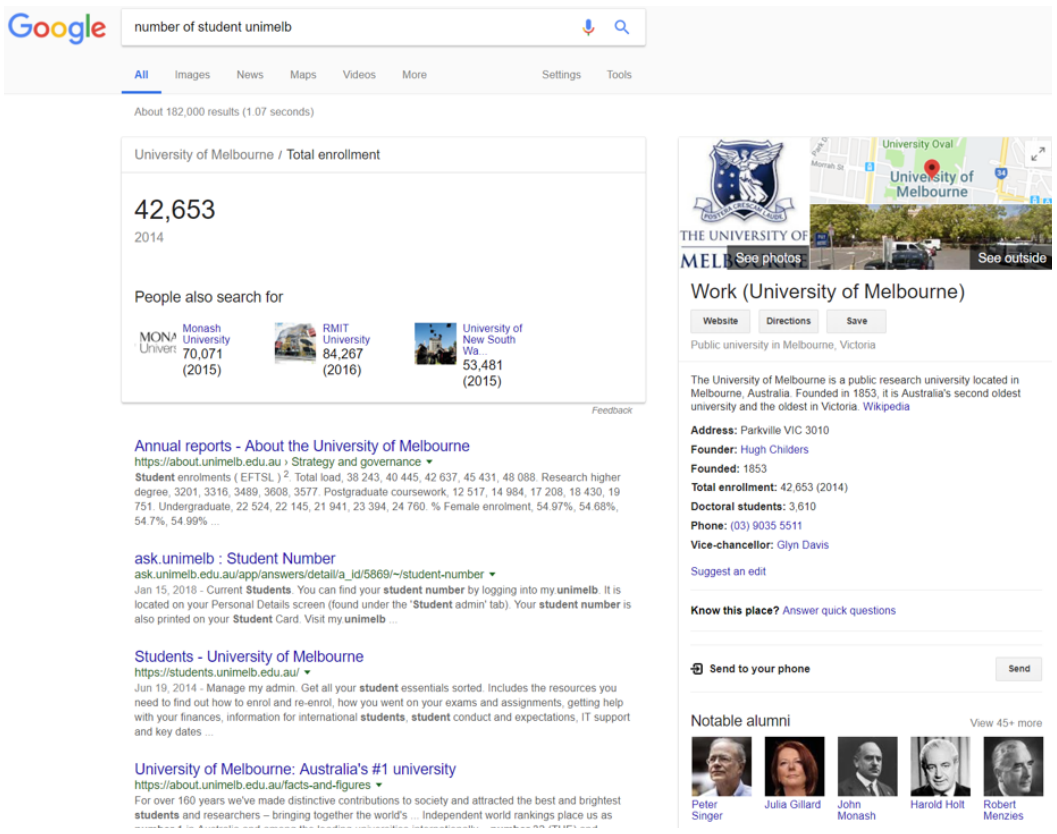
Recommendation system is a subclass of information filtering system which attempts to predict the "rating" or "preference" a user would give to an unfamiliar item. Today, recommendations provided by such systems can help users to navigate through large information spaces of product descriptions, new articles or other items. In the previous phrase of recommendation system, the most two well-known approaches of recommendation system are collaborative approach and content-based approach. Collaborative approach relies on aggregating information about users' habits or preferences, and then provides recommendations to other users according to similarity in overall preference patterns. Content-based approach aims to use supervised machine learning to derive a classifier that can distinguish between items probably to be of interest to the user and not probably to be of interest to the user. However, although these approaches seems to be practical in the real life cases, both of them own one same "ramp-up" problem, which means that both of them need a enormous number of users whose habits and preferences are known as well as a adequate number of rated items has been collected so that they can be useful for most users. This "ramp-up" problem will be solved for the appearance of knowledge base. A knowledge-based recommendation system does not rely on a base of user ratings; it does not need to collect information about a particular user for its judgements are independent of individual preference. Based on Burke (2010), a knowledge-based recommendation system even does not need to knowledge base to be prohibitively larger, since it only needs enough knowledge to evaluate whether two items are similar to each other. Graph 6-2 presents an example of a knowledge-based recommendation system.



Graph 6-2

6.3 Other Applications

Apart from QA system and recommendation system, Search engine, Chatbot, Dialogue system and Voice assistant are also the essential applications of knowledge base. Using a knowledge base as the information repository, search engine will not only provide the results directly related to the key words, but also provide the results which may be indirectly related to the key words, or results that are similar to the results of the key words. For instance, refer to graph 6-3 below, the query --”number of student unimelb” in Google will return not only the number of students in the University of Melbourne, but also show the number of students in Monash University, RMIT University and University of New South Wales in “people also search for”. Knowledge base can also be used as the data repository for Chabot, Dialogue system and voice assistance. Since these three systems all can be regarded as one type of QA system, therefore the millions of facts stored in knowledge base will highly improve the accuracy and also efficiency of those three systems.



About 182,000 results (1.07 seconds)

University of Melbourne / Total enrollment

42,653
2014

People also search for

Monash University 70,071 (2015)
RMIT University 84,267 (2016)
University of New South Wa... 53,481 (2015)

Annual reports - About the University of Melbourne
https://about.unimelb.edu.au › Strategy and governance
Student enrolments (EFTSL) 2. Total load, 38 243, 40 445, 42 637, 45 431, 48 088. Research higher degree, 3201, 3316, 3489, 3608, 3577. Postgraduate coursework, 12 517, 14 984, 17 208, 18 430, 19 751. Undergraduate, 22 524, 22 145, 21 941, 23 394, 24 760. % Female enrolment, 54.97%, 54.68%, 54.7%, 54.99% ...

ask.unimelb : Student Number
ask.unimelb.edu.au/app/answers/detail/a_id/5869/~/student-number
Jan 15, 2018 - Current **Students**. You can find your **student number** by logging into my.**unimelb**. It is located on your Personal Details screen (found under the '**Student** admin' tab). Your **student number** is also printed on your **Student** Card. Visit my.**unimelb** ...

Students - University of Melbourne
https://students.unimelb.edu.au/
Jun 19, 2014 - Manage my admin. Get all your **student** essentials sorted. Includes the resources you need to find out how to enrol and re-enrol, how you went on your exams and assignments, getting help with your finances, information for international **students**, **student** conduct and expectations, IT support and key dates ...

University of Melbourne: Australia's #1 university
https://about.unimelb.edu.au/facts-and-figures
For over 160 years we've made distinctive contributions to society and attracted the best and brightest **students** and researchers – bringing together the world's ... Independent world rankings place us as

Work (University of Melbourne)
Public university in Melbourne, Victoria

The University of Melbourne is a public research university located in Melbourne, Australia. Founded in 1853, it is Australia's second oldest university and the oldest in Victoria. Wikipedia

Address: Parkville VIC 3010
Founder: Hugh Childers
Founded: 1853
Total enrollment: 42,653 (2014)
Doctoral students: 3,610
Phone: (03) 9035 5511
Vice-chancellor: Glyn Davis

Suggest an edit

Know this place? Answer quick questions

Send to your phone

Notable alumni
Peter Singer, Julia Gillard, John Monash, Harold Holt, Robert Menzies

Graph 6-3

7. Future work and Conclusion

With the growing number of triples in large scale knowledge graphs, it has become increasingly difficult to extract route-like features for KG inference models like SFE or PRA. Enriching the knowledge graph is usually accompanied with inflating the feature space, causing substantial sparsity issues for PRA-style feature matrices, regardless of the state-of-the-art performance reported on SFE model. However, VSM are more resistant to feature inflation problems as they have a fixed low dimensional representation for entities and relations. More efforts will be put into learning a more robust embedding for knowledge graphs in the future. Recent research such as holographic embeddings (HOLE) model (Nickel et al., 2016), which learns compositional vector space representations of entire knowledge graphs based on previous TransR model, is gaining lots of advancement in prediction performance. Thus, knowledge graph embedding is considered more preferable than graph-based algorithms in future work.

Knowledge Graph Inference System is a significant component in any knowledge graph based task. This technique enables knowledge graph to reason over existing knowledge for unknown facts, as well as increases quality of known facts. It reveals the true intelligence of knowledge base that makes it different from and outperform traditional database systems. Knowledge base has opened a new era for data manipulation by converting bytes to concepts that can be understood by computers. With the combined action of evolution in Natural Language Processing and Artificial Intelligence areas, we believe that this technology will find more space to show off its utility.

8. References

Appelt, D., and Israel, C., 1999, *Introduction to Information Extraction,* viewed 24 May 2018 at <https://content.iospress.com/articles/ai-communications/aic184>

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems* (pp. 2787-2795).

Burke, R., 2010, *Knowledge-based Recommender Systems*, viewed 15 May 2018, <https://books.google.com.au/books?hl=zh-CN&lr=&id=1U_gOuKGFYYC&oi=fnd&pg=PA180&dq=recommendation+system+knowledge+base&ots=c3UeLJuUBc&sig=gT9HaLIr1hW-KFDpbsup7ZHX0CI#v=onepage&q=recommendation%20system%20knowledge%20base&f=false>

Cardie, C., 1997, *Empirical Methods in Information Extraction*, viewed 24 May, 2018 at <https://www.aaai.org/ojs/index.php/aimagazine/article/view/1322>
DBpedia.org, 2018, *Learn About DBpdia*, viewed 15 May 2018, < http://wiki.dbpedia.org/about>

Demartini, G., 2016, *A Tutorial on Leveraging Knowledge Graphs for Web Search*, viewed 24 May, 2018 at <https://link.springer.com/chapter/10.1007/978-3-319-41718-9_2>

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., ... & Zhang, W. (2014, August). Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 601-610). ACM.

Gardner, M., & Mitchell, T. (2015). Efficient and expressive knowledge base completion using subgraph feature extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 1488-1498).

Gilks, W. R., Richardson, S., & Spiegelhalter, D. (Eds.). (1995). *Markov chain Monte Carlo in practice*. CRC press.

Jurafsky, D., and Martin, J., 2016, *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition,* viewed at 24 May 2018 at <http://www.cs.colorado.edu/~martin/SLP/Updates/1.pdf>

Lao, N., Mitchell, T., & Cohen, W. W. (2011, July). Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 529-539). Association for Computational Linguistics.

Lin, Y., Liu, Z., Sun, M., Liu, Y., & Zhu, X. (2015, January). Learning entity and relation embeddings for knowledge graph completion. In *AAAI* (Vol. 15, pp. 2181-2187).

MPN.com, 2015, *YAGO*, viewed 15 May 2018, https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/

Nedellec, C., Nazarenko, A., and Bossy, R., 2016, *Information Extraction*, viewed 24 May, 2018 at <https://link.springer.com/chapter/10.1007%2F978-3-540-92673-3_30>
Nickel, M., Rosasco, L., & Poggio, T. A. (2016, February). Holographic Embeddings of Knowledge Graphs. In AAAI (pp. 1955-1961).

Nickel, M., Tresp, V., & Kriegel, H. P. (2011, June). A Three-Way Model for Collective Learning on Multi-Relational Data. In ICML (Vol. 11, pp. 809-816).

Palomares, T., 2016, *Wikipedia Knowledge Graph with DeepDive*, viewed 15 May 2018, < https://www.aaai.org/ocs/index.php/ICWSM/ICWSM16/paper/12877 >

Paulheim, H., 2016, *Knowledge graph refinement: A survey of approaches and evaluation methods*, viewed 24 May, 2018 at <https://content.iospress.com/articles/semantic-web/sw218>

Piskorski, J., and Yangarber, R., 2012, *Information Extraction: Past, Present and Future,* viewed 24 May, 2018 at <https://link.springer.com/chapter/10.1007/978-3-642-28569-1_2>

Sarawagi, S., 2008, *Information Extraction,* Foundations and Trends in Databases: Vol. 1: No. 3, pp 261-377.

Tang, J., 2008, *Information Extraction: Methodologies and Applications*, viewed 24 May, 2018 at <https://www.igi-global.com/chapter/emerging-technologies-text-mining/10174>

Wang, Z., Zhang, J., Feng, J., & Chen, Z. (2014, July). Knowledge Graph Embedding by Translating on Hyperplanes. In AAAI (Vol. 14, pp. 1112-1119).

Wöß, W. and Ehrlinger, L., 2016, *Towards a Definition of Knowledge Graphs*, viewed 24 May, 2018 at <https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Towards+a+Definition+of+Knowledge+Graphs&btnG=>