KG-Pipeline: An Automated Knowledge Graph Generation Framework

by

Alireza Vezvaei

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2022

Author's Declaration

Abstract

Knowledge Graphs (KGs) have many applications, specifically in information retrieval and question answering. Community projects are conducted for building large-scale KGs with crowdsourcing, but building KGs with this approach is costly and sometimes infeasible. Considering the rapidly growing amount of unstructured text on the Web, we highly need systems for automatic KG generation. We propose KG-Pipeline, a general-purpose end-to-end pipeline designed for automatically constructing KGs from unstructured text documents. We leverage state-of-the-art NLP models for implementing various components of the pipeline. We also utilize our generated KGs in Question Answering (QA) and evaluate the performance of our system on a QA benchmark, comparing it to previous work and an information retrieval baseline model.

Acknowledgements

I would like to express my appreciation to my supervisor, Professor Lukasz Golab, for his support, patience and guidance throughout my research.

I would like to express my sincere gratitude to Professor Mehdi Kargar, Professor Jaroslaw Szlichta, and Professor Morteza Zihayat for guiding me in my research with their patience and valuable advice.

I would also like to thank Professor Charles Clarke and Professor Olga Vechtomova who served as the readers of this thesis for their valuable time.

Last but not least, I would like to warmly thank Andrew Chai for developing an excellent demo for the system, and Mohammad Dehghan for his helpful advice and consultation.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

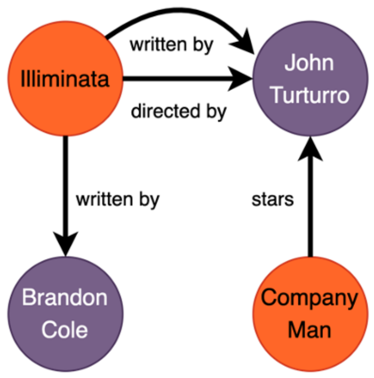
A **Knowledge Graph (KG)**[1] is a structured representation of real-world entities and the relationships between them. *(subject, predicate, object)* triples are the building blocks of KGs. Each triple represents a real-world fact as a relationship between the subject and object entities. In the graph interpretation, subject and object entities are represented as graph nodes and the predicate is represented as a labeled edge connecting subject and object nodes.

Figure 1.1 shows a part of a KG with the corresponding triples. Such triples can be extracted from text documents by applying NLP techniques that will be further discussed in the next chapters. The first three triples shown in Figure 1.1 can be extracted from the Wikipedia page of the movie *Illuminata*, and the last one can be extracted from the Wikipedia page of the movie *Company Man*, both are shown in Figure 1.2.

KGs have been proven to have many applications in both industry and academia. Google leverages a large underlying KG to improve search results with structured knowledge panels [62]. Major data-driven companies such as Amazon, Walmart, and Airbnb use their own private KGs to store, visualize, and query their customers' data. KGs are also receiving increased attention among researchers in data mining, information retrieval, and other related domains. According to a survey in [62], the most popular research applications of KGs in recent years have been *Question Answering*, *Information Retrieval*, and *Recommender Systems*.

Due to the advantages of KGs, prominent community projects are conducted for building large-scale KBs such as DBpedia [6], Wikidata [51], and ConceptNet [47, 52]. However,

[1]Also known as Knowledge Base (KB)



(a)

(Illuminata, **directed by**, John Turturro)
(Illuminata, **written by**, John Turturro)
(Illuminata, **written by**, Brandon Cole)
(Company Man, **stars**, John Turturro)

(b)

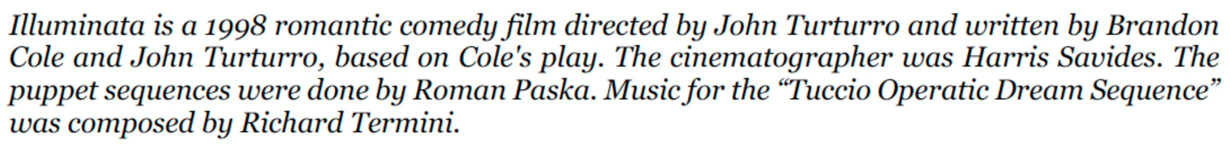
Figure 1.1: A part of a KG (a) and the triples corresponding to the KG (b)

building KBs with crowdsourcing is expensive and time-consuming; considering the rapidly growing amount of unstructured text on the Web, we need pipelines to automate KG generation.

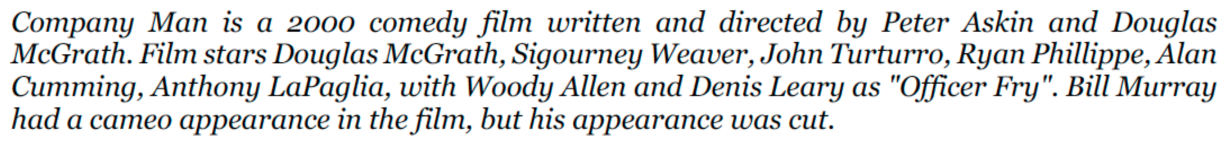
A few frameworks on automatic KG construction have been proposed in the recent years, namely **T2KG** [23], **KnowText** [8], and **AutoKG** [58]. Following the introduction of BERT in 2018, various NLP domains were subject to noticeable advances. However, as will be elaborated in the next chapters, the aforementioned systems used relatively old models, not reflecting the recent advances in NLP. Moreover, none of the systems provided either implementation details, or detailed examples of the KGs generated from input documents. In this work, we propose a general-purpose end-to-end pipeline to automatically construct KGs from unstructured text documents. Several NLP components are incorporated in the pipeline; we used state-of-the-art models to implement each of them. The implementation of each component can be easily replaced by new models in future, regardless of the other components. We plan to publish the source code after some modifications in future versions.

Question Answering (QA) is one of the main applications of the KGs. Traditional IR-based QA procedure on a text corpus includes leveraging information retrieval methods for retrieving the most relevant documents, and an answer extraction step for identifying and scoring the answer from the retrieved documents. This common procedure has two major limitations:

(a) Such systems are not capable of answering complex multi-document questions, questions requiring combining information from several sources to be answered.


Illuminata is a 1998 romantic comedy film directed by John Turturro and written by Brandon Cole and John Turturro, based on Cole's play. The cinematographer was Harris Savides. The puppet sequences were done by Roman Paska. Music for the "Tuccio Operatic Dream Sequence" was composed by Richard Termini.

(a)


Company Man is a 2000 comedy film written and directed by Peter Askin and Douglas McGrath. Film stars Douglas McGrath, Sigourney Weaver, John Turturro, Ryan Phillipe, Alan Cumming, Anthony LaPaglia, with Woody Allen and Denis Leary as "Officer Fry". Bill Murray had a cameo appearance in the film, but his appearance was cut.

(b)

Figure 1.2: The first paragraph of the Wikipedia pages of the movies *Illuminata* (a) and *Company Man* (b)

(b) Neural models used for extracting and scoring the answer are not usually explainable.

Considering the limitations of IR-based QA, KGs are preferred over plain text as the knowledge source for QA. Firstly, multi-hop inference[2] in KGs allow QA systems to answer more complex questions. Secondly, unlike IR-based QA, KG-based QA provides a KG path showing how the answer is extracted, which can be viewed as an explainability mechanism [58].

Structured data sources such as KBs are expensive and scarce compared to unstructured data sources. For the cases in which only unstructured text is available for QA, a novel alternative for traditional IR-based QA is automatically building KGs from unstructured text, then leveraging the generated KG for the QA task. To our knowledge, AutoKG [58] is the only QA system using this approach. In this work, we use a similar approach for answering questions by utilizing our automatically generated KGs. Our experiments demonstrate that our QA system performs similar to AutoKG in answering simple questions while noticeably outperforming AutoKG in answering complex questions. Moreover, we show that our QA system performs similar to a traditional IR-based baseline, while it is also explainable and capable of answering multi-document questions. To compare the performance of the models in answering multi-document questions, we introduce MDQA[3], a QA dataset consisting of questions not answerable with the content of a single document.

As an example of multi-document QA, consider the following question:

[2]Reasoning by combining the information from several (neighbouring) KG nodes

[3]Multi Document Question Answering

In which movies did the director of Illuminata act?

The question is answerable by traversing the KG in Figure 1.1: The director of *Illuminata* is *John Turturro* and *John Turturro* acted in *Company Man*. Hence, *Company Man* is a valid answer to the question. Note that in this approach, the facts extracted from both documents in Figure 1.2 are used. Thus, the question is not answerable by applying the IR-based procedure.

In summary, we make the following contributions:

- We introduce KG-Pipeline, an end-to-end framework, constituted of state-of-the-art NLP components, for automatically generating KGs from input documents.
- We developed a QA module similar to AutoKG's for answering open-domain factoid questions by leveraging our automatically-generated KGs.
- We demonstrate the performance of our QA system compared to AutoKG and an IR-based baseline on a QA benchmark. We extend the benchmark by adding a new set of question-answer pairs focusing on multi-document QA evaluation.

Chapter 2

Preliminaries

2.1 Knowledge Graphs

A *Knowledge Graph* (KG) is defined as a structured representation of real-world facts [20]. A KG consists of *nodes* representing real-world entities or concepts, and labeled *edges* representing relationships between the entities. Figure 2.1 shows a part of a KG in the domain of movies; most of the nodes represent movies and people, and most of the edges represent person-movie relations such as *directed (directed by)* and *wrote (written by)*.

Ontology is a categorization of nodes and edges of a KG. An ontology provides some background knowledge about edges and nodes, if available. For example, it can determine whether a node represents a person, a location, or a date. In general, KGs may or may not include such an ontology[1].

The term *Knowledge Base* (KB) is frequently used as a synonym for KG. However, according to [20, 52], the term *KB* is mainly referred to a set of facts stored in the format of *(subject, predicate, object)* triples (RDF[2] format) while the term *KG* emphasizes on the graph interpretation of the KBs in which the subject and the object are represented as graph nodes and the predicate is represented as a labeled edge. The KB corresponding to the KG in Figure 2.1 is demonstrated in Figure 2.2. Similar to [20], for simplicity, we will use these two terms interchangeably.

[1][55] suggested to call a KG without an ontology a *Data Graph*. However, to be consistent with most of the resources, we simply call them KG as well.

[2]Resource Description Framework

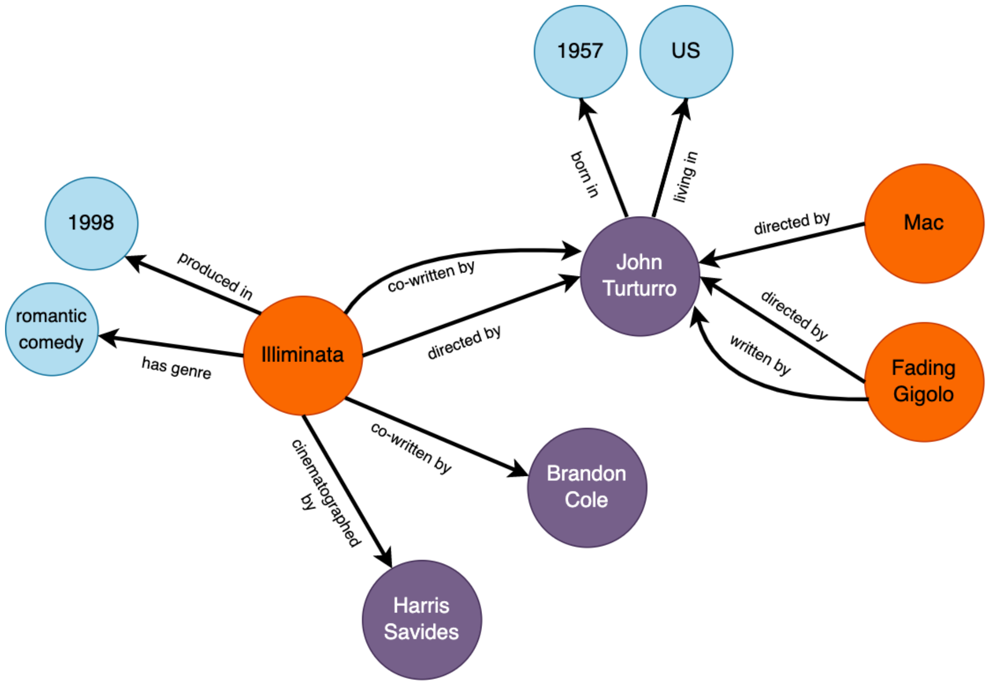


Figure 2.1: A Knowledge Graph

(Illuminata, **directed by**, John Turturro)

(Illuminata, **co-written by**, John Turturro)

(Illuminata, **co-written by**, Brandon Cole)

(Illuminata, **cinematographed by**, Harris Savides)

(Mac, **directed by**, John Turturro)

(Fading Gigolo, **directed by**, John Turturro)

(Fading Gigolo, **written by**, John Turturro)

(Illuminata, **produced in**, 1998)

(Illuminata, **has genre**, romantic comedy)

(John Turturro, **born in**, 1957)

(John Turturro, **living in**, US)

Figure 2.2: The Knowledge Base corresponding to the KB in Figure 2.1

Table 2.1: Some factoid questions and answers

Question	**Answer(s)**
the film Illuminata was written by who?	John Turturro, Brandon Cole
what was the release date of the film Illuminata?	1998
the director of Mac is also the director of which movies?	Illuminata, Fading Gigolo
which person wrote the films directed by the director of Illuminata?	John Turturro

2.2 Question Answering

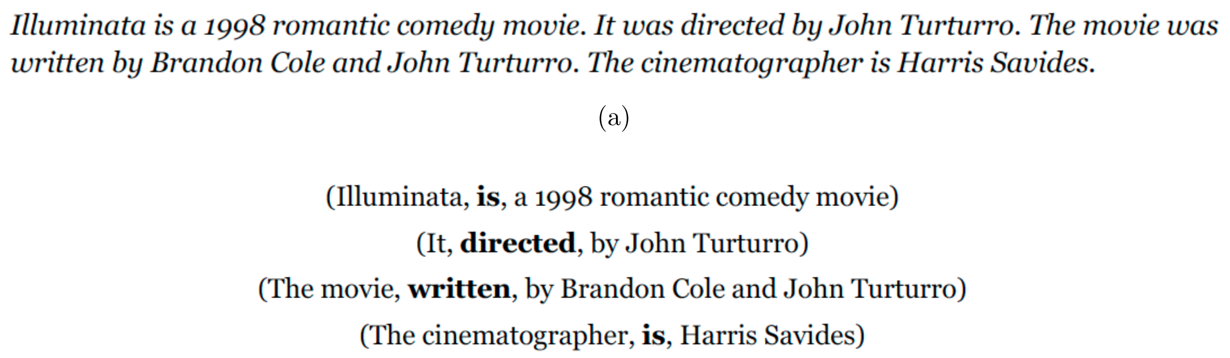
Question Answering (QA) is the task of automatically extracting answers in response to the questions asked in natural language. The QA task can be considered as an extension of *Information Retrieval* (IR) which requires a deeper level of Natural Language Understanding (NLU)[15]. While in IR we are looking for documents related to a search query, in QA tasks a precise answer in natural language is desired; thus, after retrieving a bunch of candidate documents, we need to leverage NLU models to extract the answer from the candidate documents.

Based on the domain aspect, QA systems are divided into *open domain* and *closed domain* categories[15]. Closed-domain systems are designed for answering questions in a specific domain (e.g., Medical questions) while open-domain systems are intended to be able to answer general questions in various domains. Our system is designed for open-domain QA[3].

The type and format of questions and answers vary in the QA datasets. [15, 33] categorized the question types into several groups. According to the survey conducted by [15], the most common type used in the QA datasets is *factoid* questions. Factoid questions require a single piece of information (a few words) as the answer. The answers to the factoid questions are usually real-world entities. Questions starting with *when/who/where* are examples of factoid questions [15]. In Table 2.1 some factoid questions alongside acceptable answers for each one are listed. Note that the questions in Table 2.1 are all answerable with the information represented in the KG in Figure 2.1. Our QA system is designed for answering factoid questions.

Finally, the source of knowledge that the QA systems use to extract the answer can be *structured* or *unstructured* data. Structured data include RDF KBs, SQL databases or

[3]In our experiments, we utilized *WikiMovies* which is a dataset in the movies domain. However, as will be elaborated in the next chapters, we avoided including any domain-specific components in our system.


Illuminata is a 1998 romantic comedy movie. It was directed by John Turturro. The movie was written by Brandon Cole and John Turturro. The cinematographer is Harris Savides.

(a)

(Illuminata, **is**, a 1998 romantic comedy movie)
(It, **directed**, by John Turturro)
(The movie, **written**, by Brandon Cole and John Turturro)
(The cinematographer, **is**, Harris Savides)

(b)

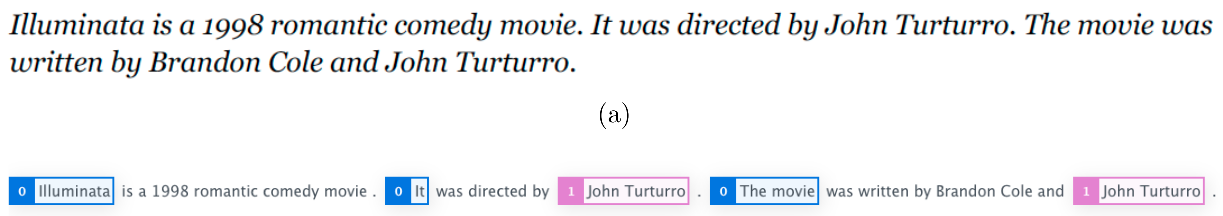
Figure 2.3: Some piece of text (a) and triples extracted by Allennlp's OpenIE model (b)

other forms of processed data whereas unstructured data is referred to corpora of plain text [15]. QA systems may use structured data, unstructured data, or a combination of the. If available, KBs and databases are more suitable for QA compared to plain text, but in many real-world scenarios, no structured source is available. Our system leverages automatically-generated KGs for QA, thus, the knowledge source is considered structured data.

2.3 Open Information Extraction

Open Information Extraction, aka OpenIE or triple extraction, is the task of extracting structured information from natural language text in the format of (subject, predicate, object) triples, similar to the RDF format in KBs. Traditional Information Extraction (IE) systems were designed to extract a limited set of target relations (predicates) in a specific domain. Such systems are not generally extensible and generalizable to be utilized in other domains [36]. On the other hand, OpenIE, introduced by [7] in 2007, is an ontology-free paradigm for extracting all types of relations from text in general domain [36, 24].

A triple extraction model takes text as input and generates a list of the triples as output. Figure 2.3 shows a piece of text alongside the triples extracted by applying AllenNLP's triple extraction implementation [3].



Illuminata is a 1998 romantic comedy movie. It was directed by John Turturro. The movie was written by Brandon Cole and John Turturro.

(a)

0 Illuminata is a 1998 romantic comedy movie . 0 It was directed by 1 John Turturro . 0 The movie was written by Brandon Cole and 1 John Turturro .

(b)

Figure 2.4: Some piece of text (a) and coreference clusters extracted by Allennlp's coreference resolution model (b)

2.4 Coreference Resolution

Coreferences (co-references) are two or more phrases referring to the same entity in a natural language text. One of the most common forms of coreferences is where a pronoun substitutes a noun phrase coming before or after it (anaphora or cataphora). For example, in both *John went to sleep when he arrived home.* and *When he arrived home, John went to sleep.* "he" and "John" are coreferences since they refer to the same entity.

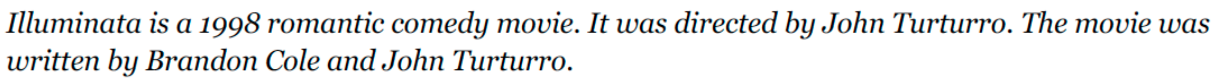
Coreference Resolution is the task of finding all the phrases (mentions) referring to the same entity in the text. A coreference resolution model takes text as input and generates a list of *coreference clusters* as output. Each coreference cluster consists of a list of mentions extracted from the text, all referring to the same entity.

Figure 2.4 shows a piece of text alongside the coreference clusters extracted by applying AllenNLP's coreference resolution model. Mentions tagged with label 0 all refer to "Illuminata" while mentions tagged with label 1 refer to "John Turturro". The AllenNLP demo is used for the visualization in Figure 2.4. [1].

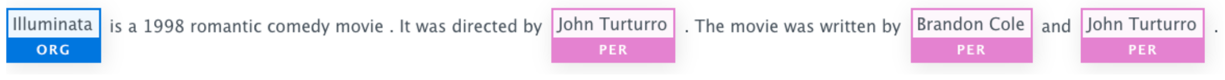
2.5 Named Entity Recognition

Named Entity Recognition (NER) is the task of identifying and categorizing named entities such as people, locations, and organizations in natural language texts. NER is one of the fundamental NLP tasks used in many higher-level NLP tasks such as coreference resolution, information extraction, and question answering [57].

NER models take text as input and identify and tag expressions belonging to named entities in the text. Figure 2.5 shows a piece of text alongside the entities tagged by


Illuminata is a 1998 romantic comedy movie. It was directed by John Turturro. The movie was written by Brandon Cole and John Turturro.

(a)


Illuminata ORG is a 1998 romantic comedy movie . It was directed by John Turturro PER . The movie was written by Brandon Cole PER and John Turturro PER .

(b)

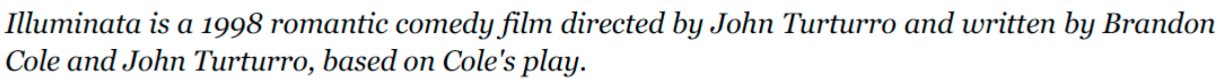
Figure 2.5: Some piece of text (a) and named entities extracted by Allennlp's NER model (b)

AllenNLP's NER model. Note that in this example, "John Turturro" and "Brandon Cole" are correctly recognized as a person while "Illuminata" is recognized as an organization, which is not accurate since it refers to a movie in this sentence. The AllenNLP demo is used for the visualization in Figure 2.5. [2].

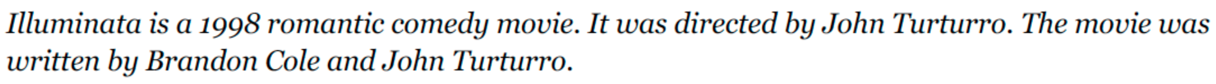
2.6 Sentence Simplification

Sentence Simplification is the task of making a sentence easier to read and understand by reducing its lexical and syntactic complexity (simplifying its vocabulary and grammar respectively) while preserving the original meaning of the sentence [4, 30]. Applying sentence simplification as a preprocessing step can help improve the results in many NLP tasks such as information extraction [4].

A sentence simplification model takes a sentence as input and generates one or multiple


Illuminata is a 1998 romantic comedy film directed by John Turturro and written by Brandon Cole and John Turturro, based on Cole's play.

(a)


Illuminata is a 1998 romantic comedy movie. It was directed by John Turturro. The movie was written by Brandon Cole and John Turturro.

(b)

Figure 2.6: A sentence before simplification (a) and after being simplified to several shorter sentences (b)

simpler sentence(s) as the output. Figure 2.6 shows a sentence and the result of applying MUSS [30] simplification model on it. It can be observed that a long sentence is broken into three grammatically simpler sentences. Also, it can be observed that the main points of the original sentence are retained while some minor details are eliminated.

Chapter 3

Related Work

As a high-level NLP task, automatic KG generation requires combining several NLP sub-tasks, which are usually implemented as an NLP pipeline including OpenIE, coreference resolution, NER, and other components [55]. In Section 3.1 the state-of-the-art models in the main NLP sub-tasks are studied. In Section 3.3, we discuss previous work on automatic KG creation. Finally, in Section 3.3, related work on KGQA is reviewed.

3.1 Related NLP sub-tasks

As will be further elaborated in the next chapter, our system consists of several components performing various NLP sub-tasks. The main tasks are sentence simplification, NER, OpenIE, and coreference resolution. we review the current state-of-the-art systems and justify our model selection for each task. Note that most of the recent NER models fulfill our need from the NER component, thus, we saw it unnecessary to conduct a literature review to justify our model selection for NER. In the rest of this section, we discuss related work in other three tasks: OpenIE, coreference resolution, and sentence simplification.

3.1.1 Open Information Extraction

From its introduction by [7] in 2007 until now, OpenIE models have had noticeable progress. The most prominent OpenIE models proposed over this period (in chronological order) are Textrunner [7], OLLIE [46], REVERB [17], ClauseIE [14], OpenIE4 [11, 37], StanfordOIE

[5], PropS [48], Graphene [10], NeuralOpenIE [13], OpenIE5 [44, 45, 38, 12], RnnOIE [49], SenseOIE [43], SpanOIE [59], IMoJIE [25], and OpenIE6 [24].

The aforementioned models used various approaches to extract accurate triples. [36] classified classic methods[1] into the four categories: (non-neural) *learning-based methods* such as Textrunner and OLLIE, *rule-based methods* such as REVERB and PropS, *clause-based methods* such as ClauseIE and StanfordOIE, and *methods capturing inter-proposition relationships* such as OpenIE4[2].

As observed by [24], recent neural-network-based systems outperformed non-neural learning-based and rule-based OpenIE systems, thus becoming the main trend in OpenIE. Authors of [24] classified recent neural models into the *labeling-based* and *generation-based* systems. Labeling-based systems, for each triple, label all the words as subject, relation, object, or none; the labeling is done independently for each triple, thus making the labeling-based methods incapable of detecting dependencies among triples. On the other hand, generation-based models use seq2seq networks to extract triples sequentially. Before extracting the next triple, generation-based models re-encode the triples extracted so far. While generation-based methods are more accurate due to the capability of capturing dependencies between triples, they are generally much slower than labeling-based approaches. Finally, OpenIE6, the state-of-the-art OpenIE model to our knowledge, bridges two methods and proposes an efficient iterative labeling framework with an accuracy comparable to generation-based methods.

Figure 3.1 demonstrates the performance of state-of-the-art OpenIE models as well as their F1 and AUC[3] scores on OpenIE benchmarks, captured from [24]. The last three rows represent the variations of OpenIE6. According to the speed column in Figure 3.1, only RnnOIE and two variations of OpenIE6 are suitable for applying to large corpora. As will be further discussed in the next chapter, we leveraged a re-implementation of RnnOIE in our framework.

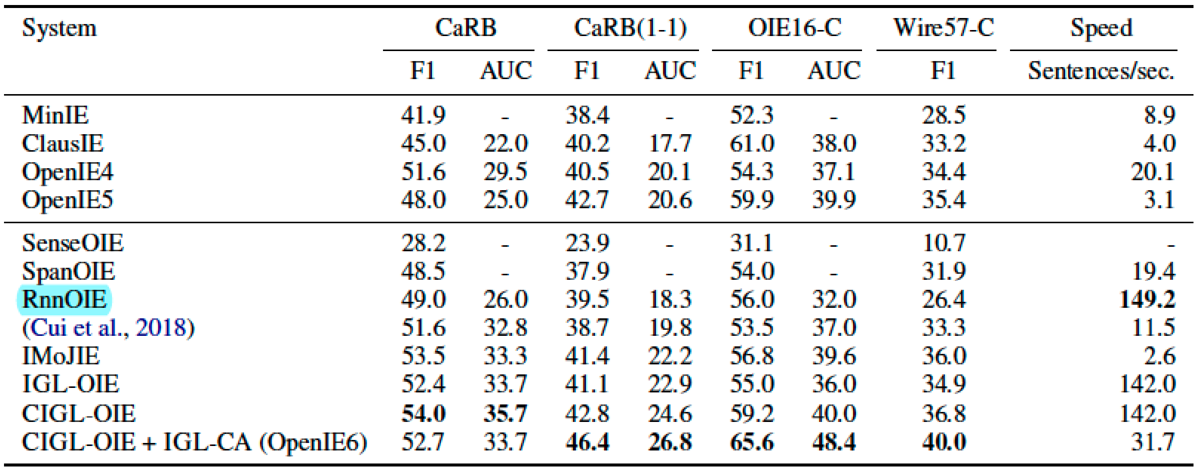
3.1.2 Coreference Resolution

Coreference resolution models demonstrated noticeable progress over recent years. [50] is a recent survey that compares the approaches and the results of state-of-the-art coreference resolution models.

[1]Models proposed before 2018

[2]For more details about each category please refer to the original paper [36].

[3]Aria Under Curve



System	CaRB		CaRB(1-1)		OIE16-C		Wire57-C	Speed
	F1	AUC	F1	AUC	F1	AUC	F1	Sentences/sec.
MinIE	41.9	-	38.4	-	52.3	-	28.5	8.9
ClausIE	45.0	22.0	40.2	17.7	61.0	38.0	33.2	4.0
OpenIE4	51.6	29.5	40.5	20.1	54.3	37.1	34.4	20.1
OpenIE5	48.0	25.0	42.7	20.6	59.9	39.9	35.4	3.1
SenseOIE	28.2	-	23.9	-	31.1	-	10.7	-
SpanOIE	48.5	-	37.9	-	54.0	-	31.9	19.4
RnnOIE	49.0	26.0	39.5	18.3	56.0	32.0	26.4	**149.2**
(Cui et al., 2018)	51.6	32.8	38.7	19.8	53.5	37.0	33.3	11.5
IMoJIE	53.5	33.3	41.4	22.2	56.8	39.6	36.0	2.6
IGL-OIE	52.4	33.7	41.1	22.9	55.0	36.0	34.9	142.0
CIGL-OIE	**54.0**	**35.7**	42.8	24.6	59.2	40.0	36.8	142.0
CIGL-OIE + IGL-CA (OpenIE6)	52.7	33.7	**46.4**	**26.8**	**65.6**	**48.4**	**40.0**	31.7

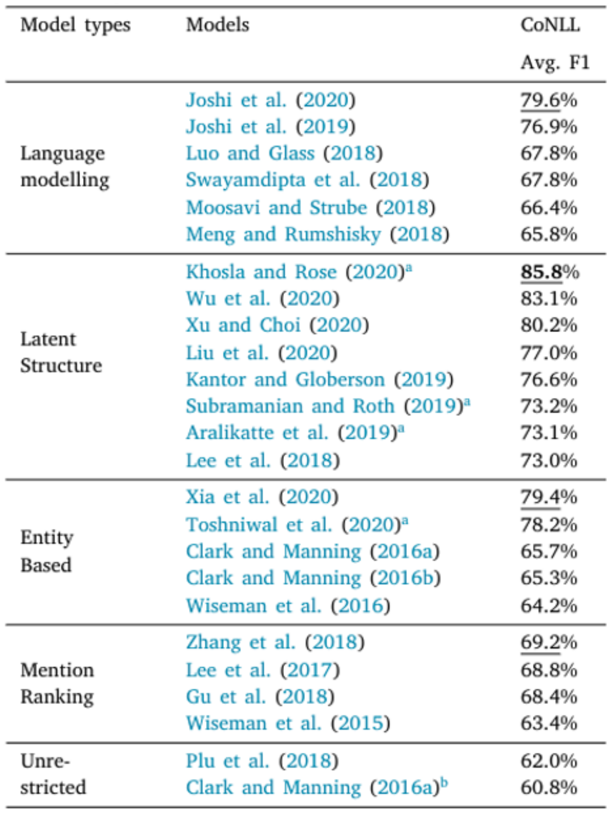
Figure 3.1: Comparison of state-of-the-art OpenIE models obtained from [24]

Figure 3.2 demonstrates the performance of state-of-the-art coreference resolution models based on average F1 on CoNLL-2012 dataset, the main benchmark in coreference resolution literature. **c2f-coref**[4] (represented by *Lee et al. (2018)* in Figure 3.2) [27] and **CorefQA**[5] (represented by *Wu et al. (2020)* in Figure 3.2.) [54] and their variations are the most popular open-sourced models to our knowledge. While *Lee et al. (2018)* used c2f-coref in combination with ELMo embeddings and achieved 73.0%, *Joshi et al. (2019)* [22] and *Joshi et al. (2020)* [21] replaced ELMo embeddings with BERT-large and SpanBERT-large embeddings and achieved 76.9% and 79.6% respectively. CorefQA also achieved as high as 83.1% as represented in Figure 3.2. As will be described in the next chapter, we leveraged a model similar to *Joshi et al. (2020)* in our framework.

3.1.3 Sentence Simplification

It has been observed that the syntactic complexity of input sentences is one of the main sources of failure in OpenIE models. More specifically, recent OpenIE models reported losing substantial recall in extracting triples from conjunctive sentences [44, 45]. To address this issue, several models have been proposed to split up a complex sentence into simpler sentences, as a preprocessing step for OpenIE.

[4]Higher-order Coreference Resolution with Coarse-to-fine Inference
[5]Coreference Resolution as Query-based Span Prediction



Model types	Models	CoNLL Avg. F1
Language modelling	Joshi et al. (2020)	79.6%
	Joshi et al. (2019)	76.9%
	Luo and Glass (2018)	67.8%
	Swayamdipta et al. (2018)	67.8%
	Moosavi and Strube (2018)	66.4%
	Meng and Rumshisky (2018)	65.8%
Latent Structure	Khosla and Rose (2020)[a]	**85.8**%
	Wu et al. (2020)	83.1%
	Xu and Choi (2020)	80.2%
	Liu et al. (2020)	77.0%
	Kantor and Globerson (2019)	76.6%
	Subramanian and Roth (2019)[a]	73.2%
	Aralikatte et al. (2019)[a]	73.1%
	Lee et al. (2018)	73.0%
Entity Based	Xia et al. (2020)	79.4%
	Toshniwal et al. (2020)[a]	78.2%
	Clark and Manning (2016a)	65.7%
	Clark and Manning (2016b)	65.3%
	Wiseman et al. (2016)	64.2%
Mention Ranking	Zhang et al. (2018)	69.2%
	Lee et al. (2017)	68.8%
	Gu et al. (2018)	68.4%
	Wiseman et al. (2015)	63.4%
Unrestricted	Plu et al. (2018)	62.0%
	Clark and Manning (2016a)[b]	60.8%

Figure 3.2: Comparison of state-of-the-art coreference resolution models obtained from [50]

Graphene proposed a rule-based framework called *discourse simplification* to split up multi-clause sentences. The framework has been shown to improve the results of state-of-the-art OpenIE models (back in 2018), such as ClauseIE and OpenIE4, when being used as a preprocessing step. Similarly, **CalmIE** proposed a coordination analyzer that splits up a complex sentence around its coordinating conjunctions (e.g. *and*, *or*, *but*) into simpler sentences which are more suitable for the downstream OpenIE. CalmIE is also shown to improve ClauseIE and OpenIE4 when being used as a preprocessing step. Finally, following CalmIE's approach, OpenIE6 proposed a novel coordination analyzer called **IGL-CA** that in combination with CIGL-OIE, the OpenIE component of OpenIE6, outperforms all the state-of-the-art OpenIE models, as demonstrated in Figure 3.1.

As suggested by [4], simplification can help improve the performance of downstream NLP tasks such as information extraction. Previous systems utilized sentence-splitting components dedicatedly designed for OpenIE. On the other hand, we found it faster and easier to leverage state-of-the-art sentence simplification models to prepare complex input sentences for downstream OpenIE. By using simplification models, we can paraphrase or split complex sentences into simpler ones rather than only splitting them around coordinating conjunctions. To our knowledge, this is the first usage of sentence simplification models in OpenIE and KG-generation.

Recent supervised and unsupervised sentence simplification models have shown promising results in simplifying the vocabulary and structure of sentences. According to the sentence simplification literature, **MUSS** is the state-of-the-art simplification model achieving the highest SARI[6] score in various simplification benchmarks [30]. As will be further elaborated in the next chapter, We leveraged MUSS as a preprocessing step in our KG-generation pipeline.

3.2 Automatic Knowledge Graph Generation

In this section, we review distinguished projects on automatic KG generation. Note that we focus on projects on generating KGs solely from unstructured text, without using any prior KG. Thus, related work on generating KGs from other sources (such as [16]) or populating existing KGs (such as [9]) are not studied.

T2KG [23] is a KG creation framework introduced in 2018. It takes unstructured text as input and applies five stages of entity mapping (NER + entity linking), coreference resolution, triple extraction, triple integration, and predicate mapping to generate a KG.

[6]The most common metric for evaluating sentence simplification models proposed by [56]

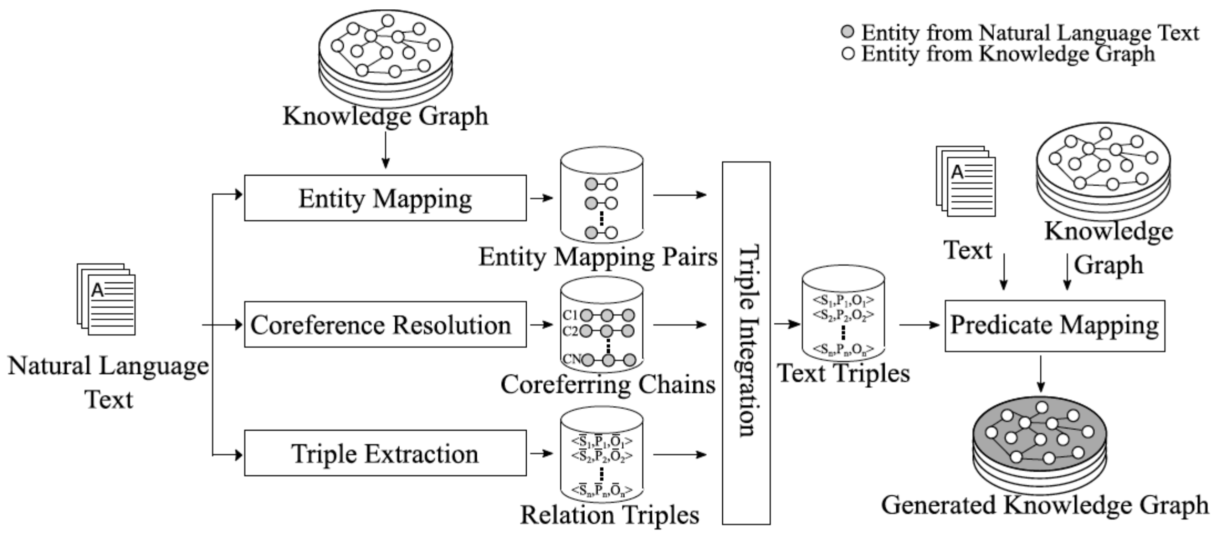


Figure 3.3: Architecture of T2KG captured from [23]

T2KG pipeline is demonstrated in Figure 3.3. The models used for OpenIE and coreference resolution are **OLLIE** [46] and **Stanford NLP tool** [26, 40], respectively. They are both relatively old[7] and far from the current state-of-the-art models in OpenIE and coreference resolution, as discussed in Section 3.1. As it can be viewed in Figure 3.3, the triple integration component combines the results of entity mapping, coreference resolution, and triple extraction to generate the triples. Then, it applies an elaborated predicate mapping module to map the extracted predicates into a set of predefined relations (ontology).

The authors of T2KG evaluated the quality of the KG creation by measuring the coverage, precision, and recall of the KB generated from 100 Wikipedia sentences (compared to a manually created KB). T2KG is not open-sourced, and the KGs constructed by T2KG are not evaluated in any downstream tasks.

Knowtext [8] is an automatic KG generation and visualization tool introduced in 2021. As demonstrated in Figure 3.4, KnowText firstly extracts triples and ontology in parallel from the input text corpus. Then, it links the entities and aligns the triples with the extracted ontology to generate the KG. Extracting triples (lower box in Figure 3.4) incorporates two bottom-level tasks of POS[8] tagging and dependency parsing as well as two main tasks including OpenIE and coreference resolution. **StanfordOIE** [5] is used for OpenIE and **neuralcoref** [34] python library is utilized for coreference resolution.

[7]Stanford NLP tool is introduced in 2011 and OLLIE is introduced in 2012

[8]part-of-speech

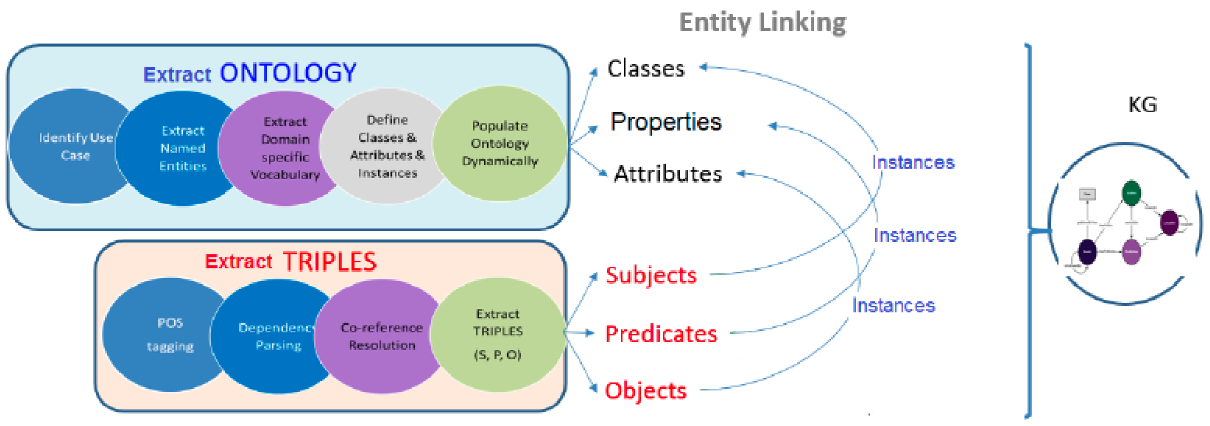


Figure 3.4: KnowText framework captured from [8]

These models are not among current state-of-the-arts in OpenIE and coreference resolution, respectively. Extracting ontology (upper box in Figure 3.4) incorporates NER and domain-specific vocabulary extraction, then binding entities to a simple 16-class ontology[9]. Note that extracting triples in KnowText is analogous to the first four steps in T2KG while extracting ontology in KnowText is similar to predicate mapping in T2KG.

KnowText is more of a commercial KG construction and visualization tool. The source code is not available, and the authors did not attempt to evaluate the generated KGs, either directly (similar to T2KG) or in a downstream application.

DELFT [61], introduced in 2020, constructs free-text KG from Wikipedia. The KG is further leveraged for factoid QA. The entities of DELFT KG are Wikipedia articles, and the edge labels are the sentences in which endpoint entities co-occurred. Unlike KnowText and T2KG which extract well-defined concise predicates, edge labels in DELFT KG are long sentences. Rather than using OpenIE to extract fine-grained relations, it preserves the entire sentences, resulting in a noisy and dirty, but high-recall KG. The reason behind this design is that DELFT builds KGs exclusively for factoid QA, hence, it prefers recall (coverage) over precision. GNNs[10] are further applied to the KGs to answer complex factoid questions. Outperforming state-of-the-art QA models, DELFT performs well in QA. However, DELFT KG is dirty and noisy, and cannot serve any purposes other than QA. The source code of DELFT is available on Github.

AutoKG [58], introduced in 2021, constructs a KG from an input document, then

[9]Some classes are: Person, Organization, Location, etc.
[10]Graph Neural Networks

applies a simple multi-hub graph traversal algorithm on the KG to answer factoid questions. For OpenIE, AutoKG utilizes OpenIE5 (combination of [44, 45, 38, 12])[11] which according to Figure 3.1, is a decent but very slow model. For coreference resolution, AutoKG uses a simple heuristic instead of well-established coreference resolution models: It encodes the extracted entities with BERT, then based on an adaptive threshold, links entities with high cosine similarity.[12]

The authors evaluated the coverage of AutoKG KGs compared to a human-annotated KB as the gold standard. Each triple of the gold standard is considered to be covered by the AutoKG KG only if the object entity is covered by the KG. As mentioned earlier, the authors also proposed a graph traversal algorithm to leverage the generated KG in QA. The details of their QA algorithm will be discussed in the next section. Performance of AutoKG in QA task is evaluated on several datasets and compared with classic IR methods such as BM25. AutoKG is not open-sourced, and some details about their KG construction and QA algorithm are unclear.

To summarize, in Table 3.1, we compared the properties of the existing models on automatic KG creation from unstructured text. As demonstrated in Table 3.1, to our knowledge, there is no open-sourced KG generation framework that takes a text corpus as input and outputs a KG that benefits developers, NLP engineers, or researchers.

3.3 Question Answering with Knowledge Graphs

In the recent decade, many systems have been proposed on open-domain factoid KBQA. A comprehensive survey on such systems is provided in [15]. According to [15] (and also [62]), two common approaches for KBQA are *Semantic Parsing* and *Information Extraction and Retrieval*. Semantic-parsing-based methods attempt to convert natural language questions into predefined logical forms such as SPARQL query language, then query the underlying KB and retrieve the answer. IE-IR-based methods, on the other hand, try to extract entities from the question, match them to the KG nodes, and retrieve the correct answer by searching in the neighbourhood of the matched KG nodes.

[11]The authors mentioned that they used CalmIE for OpenIE. CalmIE is not a standalone OpenIE model, but an extension that improves the result of a base OpenIE model. Since CalmIE is usually used in combination with OpenIE4 (constituting OpenIE5 together), we guess that the authors used OpenIE5.

[12]According to our experiments, such heuristics do not perform as well as state-of-the-art coreference resolution models; thus, it is unclear for us why the authors of AutoKG have not used a coreference resolution model instead.

Table 3.1: Comparison of automatic KG generation models

System	Automatic KG generation	Application in QA	Source-code availability	OpenIE method	Coref resolution method
T2KG	✔	✘	✘	OLLIE	Stanford NLP tool
KnowText	✔	✘	✘	StanfordOIE	Neuralcoref
DELFT	✘*	✔	✔	—	—
AutoKG	✔	✔	✘	OpenIE5	Heuristic

∗ Since each edge label in DELFT is a long sentence, it is arguable whether it can be considered as a KG or not. Anyways, it seems that the KG generated by DELFT would not be useful for any tasks other than QA.

To our knowledge, the only attempt on QA with automatically-generated KGs is made by AutoKG. Except AutoKG, all the KBQA models are designed to be used on high-quality human-annotated KGs. Automatically-generated KGs are noisier and dirtier than human-annotated KGs. Moreover, automatically-generated KGs usually lack a well-defined ontology and background knowledge about nodes, edges, and the KG structure. Therefore, it is unclear in what degree proposed KBQA models apply to automatically-generated KGs. At first glance, simple IE-IR-based methods seem more suitable for automatically-generated KGs. The reason is that semantic-parsing-based approaches require a well-defined ontology and a query language designed by human experts, which are unavailable for automatically-generated KGs. The QA approach proposed by AutoKG is a simple IR-based algorithm. The algorithm is described in the following paragraph.

AutoKG conducts an intuitive KG-traversal algorithm: It first extracts the entities from the input question and matches them with the KG nodes to obtain a list of seed nodes to start with. Candidate paths are initialized with the seed nodes, then, in an iterative manner, each candidate path is expanded by adding the neighbours of its last node. The paths are scored based on the cosine similarity of their embedding with the question embedding. Since the number of paths may grow exponentially, AutoKG conducts an expand-and-prune strategy to only keep the best k paths at each time. After several iterations, AutoKG returns the best k candidate paths that are most likely to contain the correct answer. A QA transformer can be further applied to the retrieved paths to extract the exact answer.

As the second work on QA with automatically-generated KGs, we used a QA algorithm very similar to AutoKG's. Still, some experimental study is required to examine the applicability of other types of KBQA models on automatically-generated KGs. We delegate this study to future work.

Chapter 4

Knowledge Graph Generation Pipeline

4.1 Overview

Our KG-generation framework is demonstrated in Figure 4.1. The input is a corpus of unstructured natural language text. In the first step (box no. 1), a sentence simplification module is applied to input documents as a preprocessing step. Details about the implementation of the simplification component, as well as justification for using sentence simplification, are discussed in Section 4.4. After simplification, the documents are individually processed by a pipeline of NLP components (box no. 2) including NER, OpenIE, and coreference resolution. After applying each component to each document, the results are stored alongside the document. Details about implementation of the pipeline are provided in Section 4.2. In the third step (box no. 3), entities extracted from documents are aggregated (linked) to form the KG nodes. Then, a graph generation module combines raw triples extracted in OpenIE with the results of coreference resolution and NER to attain processed triples, triples with subject and object mapped into KG nodes. Finally, edges are added between KG nodes according to the processed triples. Details of graph generation are discussed in Section 4.3. An example of executing the pipeline on an input document is provided in Figure 4.2. The input and output of the first two steps, sentence simplification and NLP pipeline, are shown in Figure 4.2. More details about the third step will be provided in Section 4.3 explaining how the KG in Figure 4.2 is generated from the extracted triples.

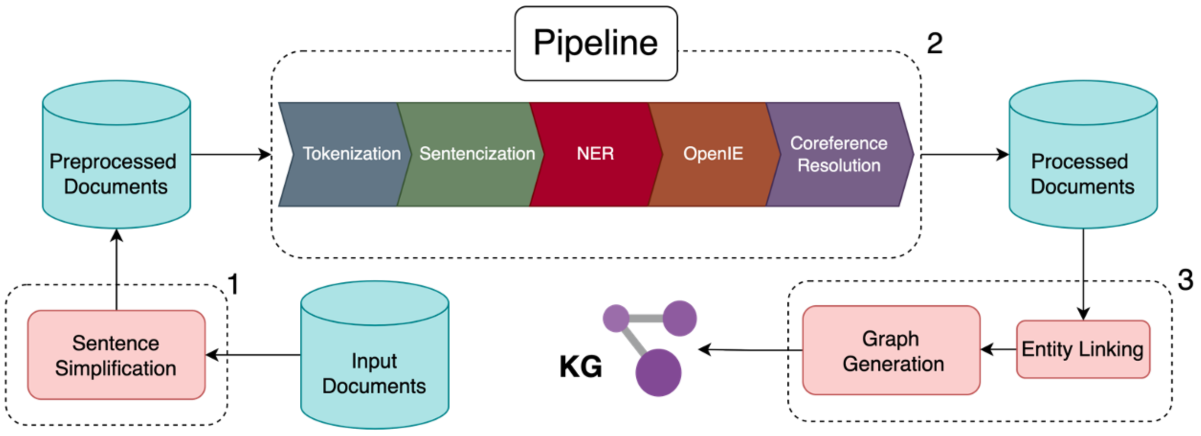


Figure 4.1: KG generation framework

4.2 NLP Pipeline

KG generation, as a demanding NLP task, requires a sequence of various NLP tasks to be applied to each document. The tasks range from bottom-level NLP such as **tokenization** and **sentencization**[1] to more advanced NLP tasks including **NER**, **OpenIE**, and **coreference resolution**. Consistent tokenization is also required to be done before applying other modules. We noticed the performance of OpenIE is superior when the input is given at sentence-level rather than document-level; that's why we used a sentencizer to split each document into sentences before applying other modules. NER is a fundamental component for KG generation. The entities extracted by the NER module are further aggregated and form the KG nodes (in the third step). OpenIE is the core of KG generation; raw triples extracted by the OpenIE module, after being processed in the third step, form the edges of the KG. Finally, coreference resolution is essential for replacing the references of entities with the original entity names in extracted triples.

We used SpaCy [19], a prominent NLP python library, to implement our pipeline. The SpaCy pipeline has many advantages including providing a default implementation for bottom-level NLP tasks (e.g., tokenization and POS tagging) and providing the capability of adding custom components. We used SpaCy default implementation for tokenization and sentencization. We also used SpaCy default components for other bottom-level tasks such as POS tagging, lemmatization, and dependency parsing; these tasks do not play a major role in KG generation but provide some useful information which is indirectly

[1] To Split up sentences

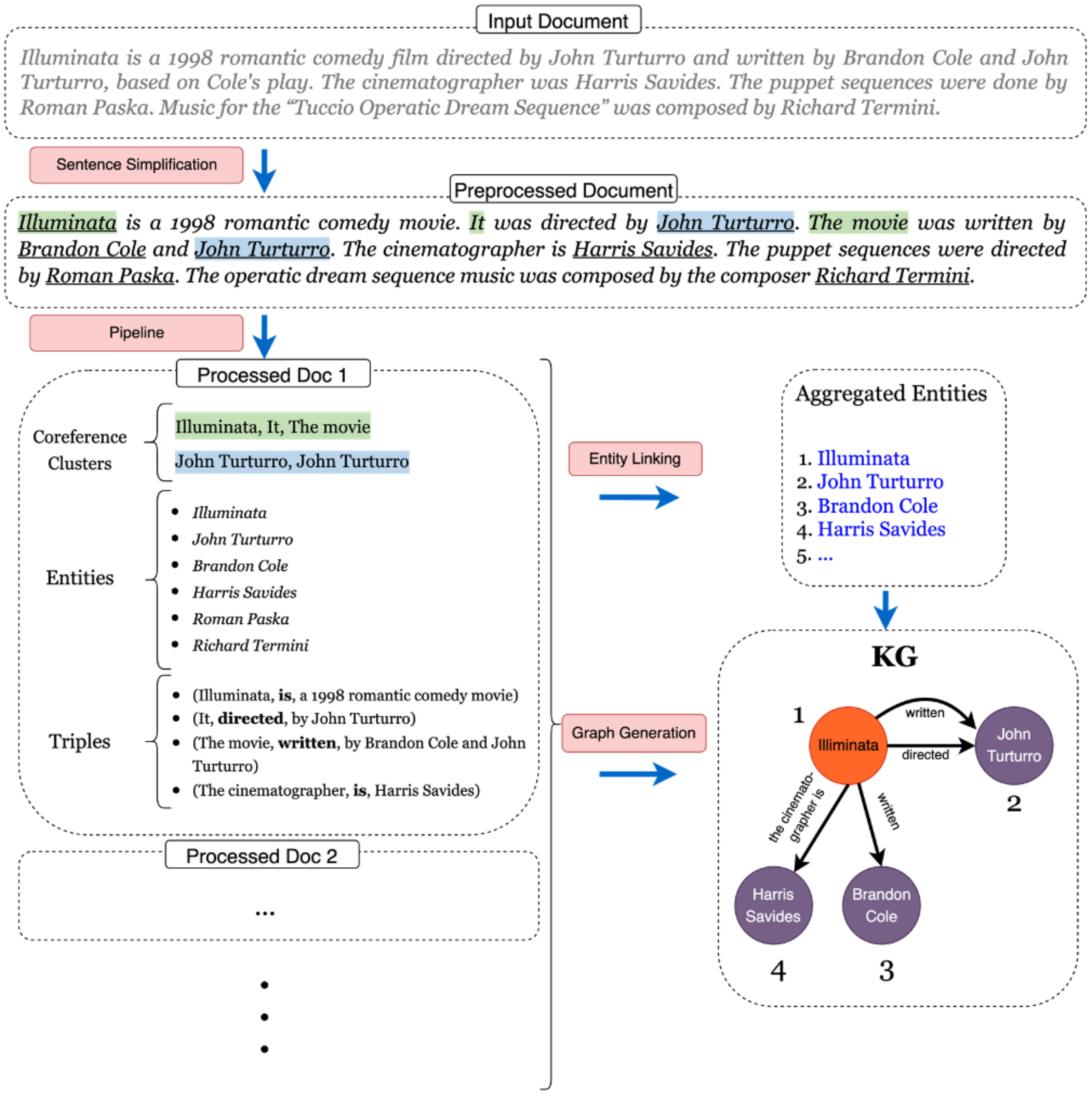

Input Document
Illuminata is a 1998 romantic comedy film directed by John Turturro and written by Brandon Cole and John Turturro, based on Cole's play. The cinematographer was Harris Savides. The puppet sequences were done by Roman Paska. Music for the "Tuccio Operatic Dream Sequence" was composed by Richard Termini.
Sentence Simplification
Preprocessed Document
Illuminata is a 1998 romantic comedy movie. It was directed by John Turturro. The movie was written by Brandon Cole and John Turturro. The cinematographer is Harris Savides. The puppet sequences were directed by Roman Paska. The operatic dream sequence music was composed by the composer Richard Termini.
Pipeline
Processed Doc 1
Coreference Clusters
Illuminata, It, The movie
John Turturro, John Turturro
Entities
Illuminata
John Turturro
Brandon Cole
Harris Savides
Roman Paska
Richard Termini
Triples
(Illuminata, is, a 1998 romantic comedy movie)
(It, directed, by John Turturro)
(The movie, written, by Brandon Cole and John Turturro)
(The cinematographer, is, Harris Savides)
Processed Doc 2
...
Entity Linking
Graph Generation
Aggregated Entities
1. Illuminata
2. John Turturro
3. Brandon Cole
4. Harris Savides
5. ...
KG
1
Illiminata
written
directed
John Turturro
2
the cinemato-grapher is
written
Harris Savides
Brandon Cole
4
3

Figure 4.2: An example of running KG generation framework on a sample input document

used in the third step. For OpenIE, NER, and coreference resolution, we implemented our custom components by leveraging state-of-the-art models, as specified in the rest of this section. The advantage of using such a pipeline is that the components are implemented independently, thus, the implementation of each component can be easily replaced by a newer model.

SpaCy uses a `Doc` object to store each document. After applying each component, the results are stored as an attribute of the corresponding `Doc` object. More specifically, in case of our custom components, extracted entities are stored in `Doc._.entities` after NER, triples are stored in `Doc._.triples` after OpenIE, and coreference clusters are stored in `Doc._.clusters` after coreference resolution.

In the rest of this section, models used for NER, OpenIE, and coreference resolution are introduced.

4.2.1 Open Information Extraction

Scalability was the main criteria in selecting the OpenIE model since the model should be applied to all sentences of large datasets. As mentioned in Section 3.1, among the state-of-the-art OpenIE models, only RnnOIE and OpenIE6 (without the coordination analyzer) are efficient enough for our task. OpenIE6 is open-sourced and available on Github as a python package, while RnnOIE is re-implemented by AllenNLP and is provided within the **allennlp** python library, making it easier to use as a component of NLP pipelines [18]. Taking software engineering aspects of the system into account, we decided to utilize AllenNLP's implementation of RnnOIE which is efficient, easy to use, and consistent with other components. Note that as discussed in Section 4.4, we combined RnnOIE with a sentence simplification model to improve the accuracy of OpenIE. Hence, the resulting system is expected to perform better than the scores reported in Figure 3.1.

4.2.2 Coreference Resolution

As discussed in Section 3.1, c2f-coref (SpanBERT-large implementation) and corefQA are the two open-sourced state-of-the-art coreference resolution models. c2fcoref is implemented by AllenNLP and provided within the **allennlp** python library, making it much easier to use [18]. Note that in AllenNLP's implementation of c2f-coref, ELMo embeddings are substituted by SpanBERT embeddings. Thus, it is expected to perform similar to *Joshi et al. (2020)*, the first model in Figure 3.2, achieving the second best score among all the open-sourced models to our knowledge. We leveraged the AllenNLP model which is efficient, easy to use, and consistent with other components of the pipeline.

4.2.3 Named Entity Recognition

Most of the recent NER models fulfill our need from the NER component since the only information we use from NER output is whether each span is an entity or not (we do not use predicted entity types). Thus, the NER component is not critical in our pipeline in the sense that a decent NER model fulfills our requirement. We simply used AllenNLP's NER model [18] which is easy to use and consistent with other components. The AllenNLP's NER model is based on the model proposed in [39].

4.3 Entity Linking And Graph Generation

After processing documents in the pipeline, the entities extracted from documents are filtered, aggregated, and deduplicated. For filtering entities, some simple rules are applied. For instance, if one of the entities extracted from a document is a substring of another one, the former will be eliminated. The reason is that in many sentences, a shortened coreference to a previously-mentioned entity is mistakenly recognized as a different entity. In the following example, *Christopher Nolan* and *Nolan* are both recognized as entities. However, after applying the aforementioned rule, only the former one is kept, as desired.

"*Batman Begins is a 2005 British-American superhero film based on the fictional DC Comics character Batman, co-written and directed by Christopher Nolan ... After a series of unsuccessful projects to resurrect Batman on screen following the 1997 critical failure of 'Batman & Robin', Nolan and David S. Goyer began to work on the film ...*"

In addition to entity filtering, some domain-specific entities (which are not already recognized by the NER) can be added in this step. For example, in the movie domain, genres and production years can be added to the list of entities; genres like *comedy* and years like *1997* are not normally recognized as named entities. However, recognizing them as graph nodes might be desired in some applications. Specifically, adding them can improve the QA results, when the KG is leveraged for the QA task.

After refining entities, the entities are aggregated and linked. In this step, similar entities that are most likely to refer to an identical entity are merged and deduplicated. After filtering, aggregating, and linking, the resulting set of entities will constitute the the KG nodes.

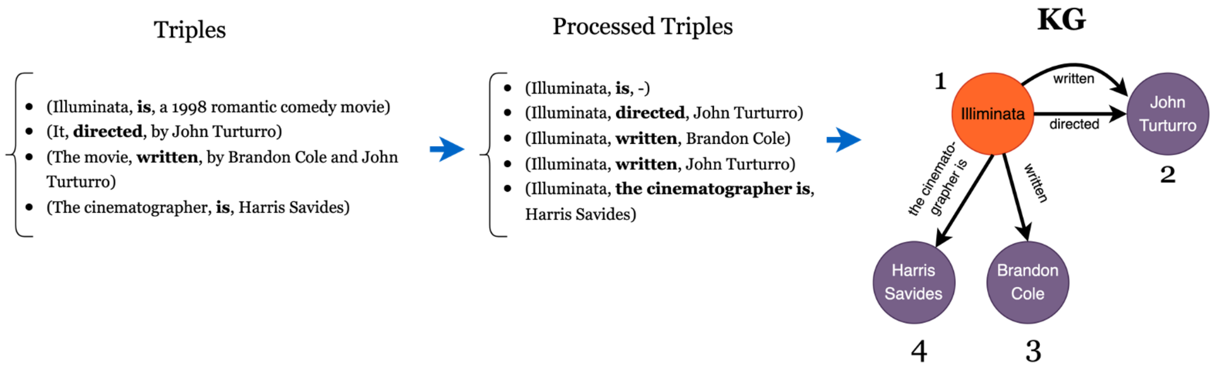

Triples
(Illuminata, **is**, a 1998 romantic comedy movie)
(It, **directed**, by John Turturro)
(The movie, **written**, by Brandon Cole and John Turturro)
(The cinematographer, **is**, Harris Savides)
Processed Triples
(Illuminata, **is**, -)
(Illuminata, **directed**, John Turturro)
(Illuminata, **written**, Brandon Cole)
(Illuminata, **written**, John Turturro)
(Illuminata, **the cinematographer is**, Harris Savides)
KG
1
Illiminata
written
directed
John Turturro
2
the cinemato-grapher is
written
Harris Savides
Brandon Cole
4
3

Figure 4.3: Graph generation part of the example in Figure 4.2.

The final step is graph generation. In this step, the triples extracted from each document are processed and converted to KG nodes. For each subject (of a triple), if some part of the subject or a coreference of that is recognized as a named entity, the subject is linked to the corresponding entity node. The exact same procedure is done for the objects of the extracted triples. Then, for triples having both subject and object linked to KG nodes, an edge labeled with the predicate will be added between subject and object nodes. Triples lacking subject/object nodes will be stored as attributes of the object/subject node. Some rule-based heuristics are also applied to retrieve the correct subject/object node when the subject/object is not linked to a KG node. The triples in which none of the subject or object are linked into a KG node are thrown away.

Figure 4.3 shows the graph generation part corresponding to the example demonstrated in Figure 4.2. *It* in *(It, directed, by John Turturro)* is replaced by *Illuminata*, which is coreference with *It* and is also recognized as a named entity. Also, *by John Turturro* is replaced by *John Turturro* which is recognized as a named entity. The resulting triple will be *(Illuminata, directed, John Turturro)* that leads to drawing an edge labeled with *directed* between nodes of *Illuminata* and *John Turturro* in the KG. Similarly, *(The movie, written, by Brandon Cole and John Turturro)* will be processed, resulting in *(Illuminata, written, Brandon Cole)* and *(Illuminata, written, John Turturro)*.

In the case of *(Illuminata, is, a 1998 romantic comedy movie)*, the first extracted triple, the object does not link to any nodes, unless we intervene in NER in order to recognize domain-specific words such as *comedy* as named entities. In this case, an edge labeled with *is* would be added between the nodes representing *Illuminata* and *comedy*. Otherwise, the sentence *"s a 1998 romantic comedy movie"* is stored as an attribute of *Illuminata*, without adding any edges to the graph. To keep our framework domain-agnostic, we decided to conduct the latter approach which is avoiding any intervention in NER, and only storing *comedy* as an attribute of *Illuminata*.

Finally, in the case of *(The cinematographer, is, Harris Savides)*, the last triple, the subject is not initially linked to a node, however, a heuristic is applied to recognize *Illuminata* as the subject node; knowing that the context (the title of the document) is *Illuminata*, we set the subject node of such subjectless triples to *Illuminata* and recognize the combination of subject and predicate as the new predicate. The resulting triple would be *(Illuminata, the cinematographer is, Harris Savides)* which is shown in the KG.

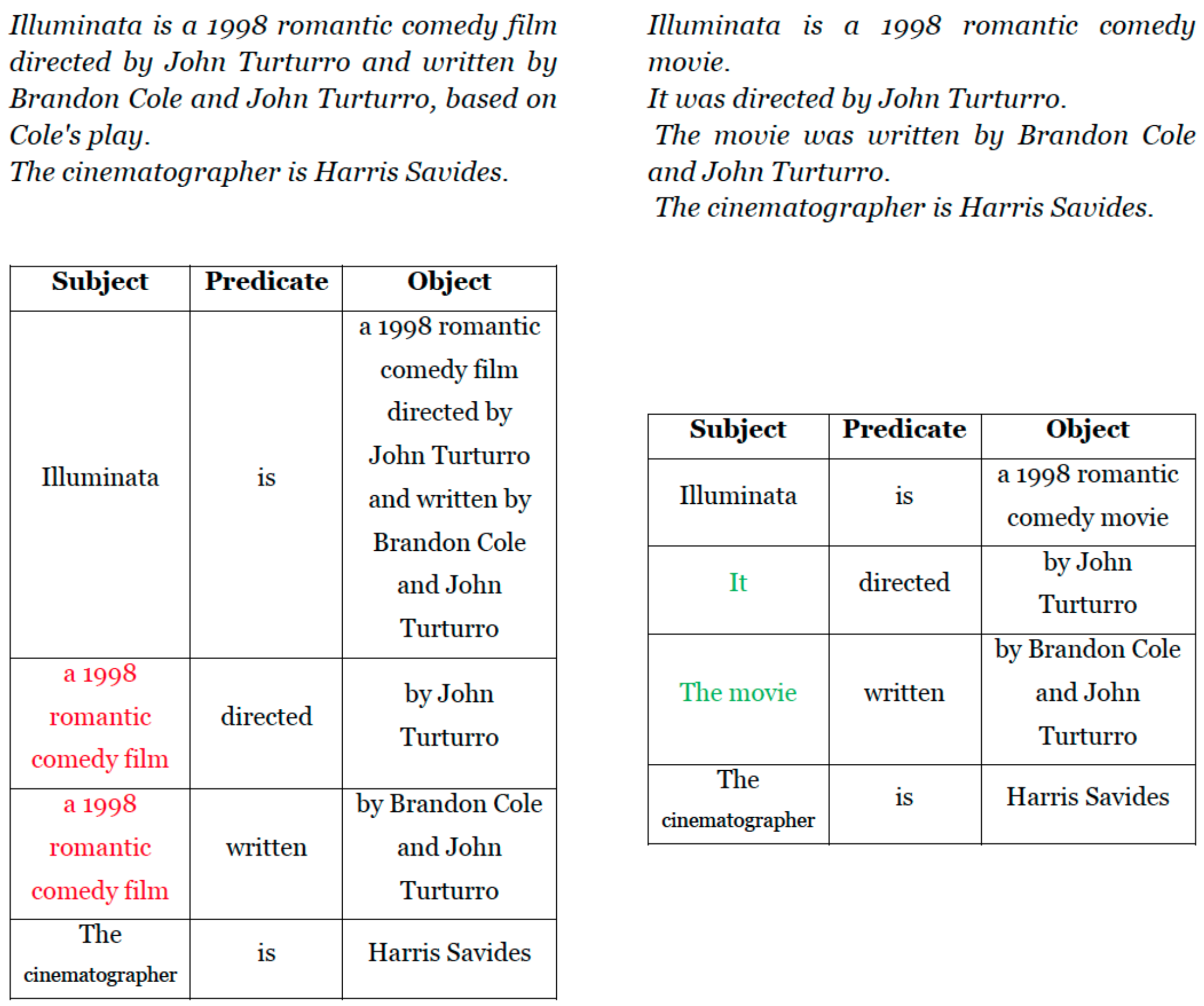
Note that while some previous work such as T2KG map the extracted predicates to a set of predefined relations, we do not apply any predicate mapping (ontology mapping). The reason is that thousands of unique predicates are extracted from a text corpus to build a general-purpose KG; mapping all the predicates to a limited set of relations leads to losing a large amount of information, as a study in [23] reported that 23% of KG generation errors arose from predicate mapping. Specifically, complicated predicates, such as *'the cinematographer is'*, will be lost in predicate mapping. Although predicate mapping standardizes the KG and makes it evaluable, the KGs generated without predicate mapping cover more facts and are more powerful in IR and QA tasks. Hence, we decided not to apply predicate mapping in our framework.

4.4 Sentence Simplification

We noticed that the syntactic complexity of sentences prevents OpenIE models from correctly extracting the triples. As described in Section 3.1, we leverage MUSS for breaking complex sentences into the simpler ones[2]. As demonstrated in Figure 4.1, simplification is the first step in the framework; it takes raw documents as input and outputs simplified (preprocessed) documents which are ready to be processed by the pipeline.

MUSS is built on top of **Access** [31], a sentence simplification model providing four control tokens for adjusting the simplification according to the application. While simplifying the syntactic structure of sentences is beneficial for downstream OpenIE, simplifying the vocabulary is unnecessary and potentially harmful. Therefore, we tune control tokens in such a way that maximizes syntactic simplification while minimizing lexical simplification. Specifically, by setting the target ratio of *WordRankRatioPreprocessor* token to 1, we prevented the model from changing the vocabulary of the original sentences.

[2]Note that despite some state-of-the-art OpenIE models, such as OpenIE6, have built-in coordination analyzer, RnnOEI lacks a built-in sentence simplification component. Hence, we need to add the simplification as a preprocessing step before OpenIE



Illuminata is a 1998 romantic comedy film directed by John Turturro and written by Brandon Cole and John Turturro, based on Cole's play.
The cinematographer is Harris Savides.

Subject	Predicate	Object
Illuminata	is	a 1998 romantic comedy film directed by John Turturro and written by Brandon Cole and John Turturro
a 1998 romantic comedy film	directed	by John Turturro
a 1998 romantic comedy film	written	by Brandon Cole and John Turturro
The cinematographer	is	Harris Savides

Illuminata is a 1998 romantic comedy movie.
It was directed by John Turturro.
The movie was written by Brandon Cole and John Turturro.
The cinematographer is Harris Savides.

Subject	Predicate	Object
Illuminata	is	a 1998 romantic comedy movie
It	directed	by John Turturro
The movie	written	by Brandon Cole and John Turturro
The cinematographer	is	Harris Savides

Figure 4.4: Results of applying AllenNLP's OpenIE on a piece of text before simplification (left) and after simplification (right)

Figure 4.4 demonstrates the advantage of using sentence simplification on a sample input. It can be observed that in the simplified case, subjects and objects are accurate and fine-grained. On the othe hand, in the non-simplified case, long conjunctive sentences are identified as the subject or object of a single triple. More specifically, notice that in the simplified version, *It* and *The movie*, the subjects of the second and third sentences, both belong to the coreference cluster of *Illuminata*. Therefore, after combining the results of NER, OpenIE, and coreference resolution, the final set of triples would include *(Illuminata, directed, Joun Turturro)*, *(Illuminata, written, Brandon Cole)*, and *(Illuminata, written, John Turturro)*. On the other hand, in the non-simplified case, *a 1998 romantic comedy film* is the subject of both the second and third triples. None of the words in the subject span belong to any coreference cluster or named entity span. As a result, the second and third triples would be thrown away without extracting any of the three important triples mentioned earlier.

Chapter 5

Question Answering via KG Utilization

5.1 Overview

As mentioned in Section 3.3, AutoKG [58] uses a simple algorithm consisting of extracting seed entities from the input question and KG-traversal for finding paths relevant to the input question. We utilized a QA algorithm similar to AutoKG's description for retrieving the candidate paths. Moreover, we extend AutoKG's framework by adding an answer extraction component. While AutoKG returns candidate paths with no attempt to further process them, our answer extraction component extracts the exact answer from the candidate paths.

Our QA framework is demonstrated in Figure 5.1. In the first step, similar to AutoKG, seed entities are extracted from the input question and mapped into the KG nodes. Details of this step are discussed in Section 5.2. In the second step, explained in Section 5.3, we search the neighbourhood of the seed nodes to find paths that are most likely to contain the correct answer to the input question. This step is also a re-implementation of the KG-traversal described in AutoKG. Finally, we apply a state-of-the-art neural QA model to extract the answer from the candidate paths. This step completes our QA framework and allows us to compare our QA results with other QA methods. More details on the third step are discussed in Section 5.4.

Figure 5.2 demonstrates the QA procedure on an input question. Firstly, the word *Illuminata* is extracted from the question as the seed entity, and then mapped to the

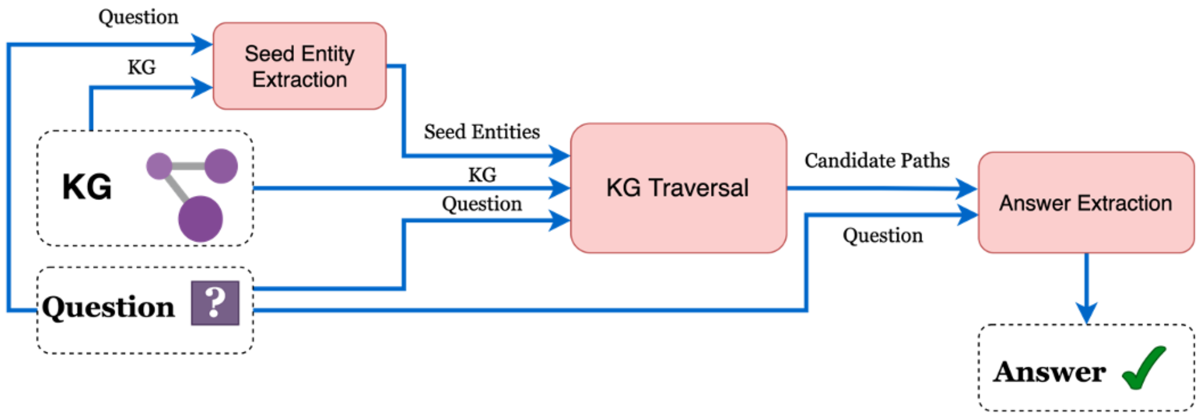


Figure 5.1: The QA Framework

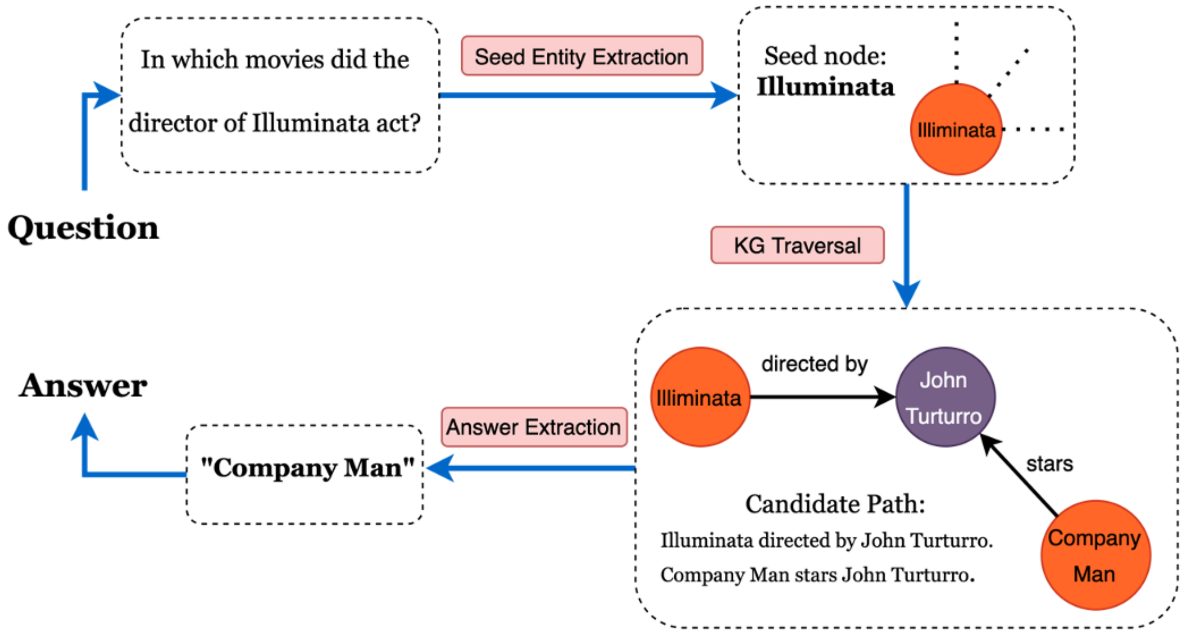


Figure 5.2: An example of the QA Framework

KG node representing *Illuminata*. In the second step, among short paths starting from the *Illuminata* node, the path containing *Illuminata*, *John Turturro*, and *Company Man* is identified as a candidate path potentially containing the answer. The text corresponding to this path is *"Illuminata directed by John Turturro. Company Man stars John Turturro"*. In the final step, a QA model takes the question and the text (corresponding to the candidate path) and extracts the answer which is *Company Man*.

5.2 Seed Entity Extraction

In this step, firstly, the same NER model as the one used in KG-generation is used to extract entities (called seed entities) from the input question. In addition to NER, several simple heuristics are applied to improve the accuracy of entity recognition. The heuristics are either designed for i. Identifying phrases that are likely to represent an entity, but are not recognized by the NER model (such as title-case phrases). ii. Filtering out entities recognized by the NER model that are not likely to represent actual entities (such as entities that are substrings of longer entities).

After entity extraction, extracted entities are mapped into the KG nodes. If an exact match is not found, some similarity metrics are used to map the entities to the most similar KG nodes. The mapped nodes are called seed nodes since the KG-traversal in the next step will be initialized with these nodes. In the QA datasets we used for evaluation, almost all the questions only contain one seed node similar to the input question in Figure 5.2. However, the algorithm theoretically works for questions with multiple entities, as well.

5.3 KG Traversal

In this step, having the input question, KG, and a set of starting nodes, we traverse the KG to find paths containing the answer. Algorithm 1 is a high-level pseudocode describing the KG-traversal procedure. The pseudocode is analogous to the pseudocode provided by AutoKG [58] while differing in details and terminologies. The main inputs are the KG, seed nodes, and the input question. Other inputs include *hops*, a hyper-parameter specifying the maximum depth of graph search, and *beam_size*, another hyper-parameter specifying the maximum number of active paths that are kept to be expanded in future rounds of the algorithm.

In the beginning, *active_paths* is initialized with seed nodes. Then in *hops* rounds, the paths are expanded with a breadth-first expand-and-prune strategy. In each round, active

Algorithm 1 KG Traversal algorithm

```
    Input: $KG$, $seed\_nodes$, $question$, $hops$, $beam\_size$
    Output: $candidate\_paths$
1: $active\_paths \leftarrow seed\_nodes$
2: for $i = 0 \rightarrow hops$ do
3:     $expanded\_paths \leftarrow \{\}$
4:     for each $path \in active\_paths$ do
5:         $head \leftarrow$ last node of $path$
6:         for each $new\_node \in Neighbours(head)$ do
7:             $expanded\_path \leftarrow path + [new\_node]$
8:             add $Text(expanded\_path)$ to $candidate\_paths$
9:             add $expanded\_path$ to $expanded\_paths$
10:    sort $expanded\_paths$ according to their similarity to $question$
11:    $active\_paths \leftarrow$ The first $beam\_size$ paths in $expanded\_paths$ having highest scores
12: return $candidate\_paths$
```

paths are expanded by the neighbours of their front node (line 7). The text corresponding to each expanded path (the result of concatenating consecutive nodes and edge labels along the path) is added to the list of candidate paths (line 8). Expanded paths are sorted based on the similarity of their corresponding text to the input question (line 10). **all-MiniLM-L6-v2**, a popular sentence-similarity model, is utilized for similarity calculation. The most relevant paths are used as the active paths for the next rounds while less relevant paths are pruned away (line 11). As mentioned in [58], the pruning is due to an exponential growth in the number of active paths, making it impossible to investigate all the paths. Finally, the list of candidate paths (in the text format) is returned to be further processed in the answer extraction step.

Back to the example shown in Figure 5.2, the KG-traversal begins with *Illuminata* as the seed node. In the first round, it is expanded with the neighbouring nodes, including *John Turturro*. The resulting path, {*Illuminata*, *John Turturro*}, is kept in the list of active paths since its corresponding text, *"Illuminata directed by John Turturro"*, has a relatively high similarity with the input question. In the next round, the path is expanded with *Company Man*, resulting in {*Illuminata*, *John Turturro*, *Company Man*}. The text form of this path (*"Illuminata directed by John Turturro. Company Man stars John Turturro."*) is appended to the list of candidate paths as well as other promising candidates. The answers will be extracted from candidate paths in the next step.

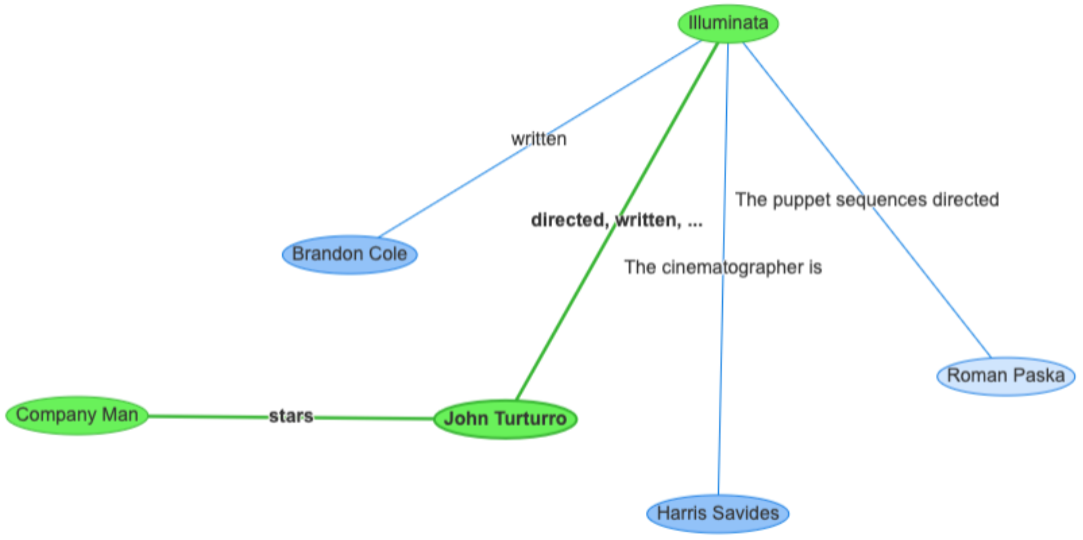


Figure 5.3: A screenshot of our QA demo corresponding to the example in Figure 5.2

5.4 Answer Extraction

While AutoKG ends up reporting the most relevant candidate paths, we further process candidate paths to extract and report exact answers. For answer extraction, We leverage **roberta-base-squad2** (RoBERTa-base model [29] fine-tuned on SQuAD2.0 dataset [41] provided by Hugging face transformers python library [53]), a popular QA model. It takes the question and context as input and extracts the answer from the context along with a confidence score. We give each candidate path separately to the model as the context. For each candidate path, we store the extracted answer as well as the confidence score. The answers with the highest confidence scores (along with their corresponding paths) are then reported as the final answers.

Returning to the example in Figure 5.2, the QA model takes candidate paths as context, trying to find a confident answer to *In which movies did the director of Illuminata act?*. Taking *"Illuminata directed by John Turturro. Company Man stars John Turturro."* as the context, it extracts *Company Man* with a relatively high score. Therefore, *Company Man* is returned as the answer and the path from *Illuminata* to *Company Man* is returned as the path leading to the answer.

A screenshot of our QA demo is demonstrated in Figure 5.3. The question *"In which movies did the director of Illuminata act?"* is entered into the demo. The system applies the QA procedure, retrieves the best answers, and visualizes the paths containing the best retrieved answers.

Chapter 6

Experiments

6.1 Overview

In this section, we aim to evaluate the KGs generated by our pipeline. Among KnowText, T2KG, and AutoKG, the three KG generation frameworks described in Section 3.2, the former did not attempt to evaluate the generated KGs. T2KG manually generated a gold standard KB from 100 Wikipedia sentences and evaluated the precision and recall of the automatically-generated KG against the gold standard. Since T2KG applies predicate mapping, the KGs generated by T2KG are evaluable by this approach. On the other hand, calculating the precision and recall of the KGs generated by our pipeline is not straightforward. The reason is that without predicate mapping, predicates may appear in various forms, making it difficult to decide whether a predicate should be considered correct or incorrect.

As an example, consider the triples that might be extracted from *"Illuminata is a 1998 romantic comedy film directed by John Turturro"*. While the subject node is *Illuminata* and the object node is *John Turturro*, extracted predicate might be various phrases such as *'directed'*, *'is directed'*, *'is'*, *'directed by'*, *'film directed by'*, *'is a 1998 romantic comedy film directed by'*, etc., depending on the underlying OpenIE method. Clearly, predicates such as *'directed'* and *'directed by'* are acceptable while predicates like *'is'* and *'is a 1998 romantic comedy film directed by'* are not acceptable edge labels for a KG. Even though we have access to a gold standard KB, automatically deciding whether each predicate matches the gold standard or not is not straightforward.

AutoKG indirectly evaluated the KGs, combined with a KG-traversal algorithm, in question answering. The authors also conducted a direct evaluation by calculating the coverage (recall) of the retrieved triples against a gold standard KB. Similar to our framework, AutoKG does not apply predicate mapping, thus, the authors needed a way to overcome the aforementioned issue with calculating recall. The authors considered each triple of gold standard KB to be covered by the generated KG only if the object entity is covered by the KG. This way, they did not need to take predicates into account. However, this method of reporting coverage seems simplistic and misleading to us since it does not evaluate the quality of extracted predicates.

As discussed above, we are not aware of a fair way of directly evaluating KGs. Hence, similar to AutoKG, we indirectly evaluate the KGs in question answering as the main downstream application of KGs. This way we can easily quantify the results and compare KGs with each other. In the rest of this chapter, we present our QA experiments. Datasets used in our experiments are introduced in Section 6.2. Evaluation approaches and metrics are discussed in chapter 6.3. In Section 6.4, we discuss the configurations of the system we used for our implementations as well as the hyper-parameter settings. Finally, in Section 6.5 we discuss the baseline models and compare the QA results of KG-Pipeline with the baselines.

6.2 Datasets

Similar to [58], we evaluated our QA algorithm on two QA benchmarks: **WikiMovies** [32] and **MetaQA** (2-hop and 3-hop) [60]. Both datasets consist of Q&A pairs from the same corpus: 18128 Wikipedia articles in the movie domain. Each document includes text extracted from the Wikipedia page of a movie. Documents corresponding to the movies *Illuminata* and *Company Man* are shown in Figure 1.2. We also introduced **MDQA**, a QA dataset consisting of multi-document questions: questions which require combining information from multiple documents. Although the three datasets are extracted from the same corpus, they include different question types. In the rest of this section, we discuss the details of the datasets.

6.2.1 WikiMovies

WikiMovies, introduced in 2016, includes 11 types of questions. The question types are represented in Table 6.1. In terms of KGQA, all questions in WikiMovies are 1-hop; the answer is always available in the neighbourhood of the seed node in an ideal KG. The dataset is partitioned into training, development, and testing sets. The testing partition includes 9952 Q&A pairs.

Table 6.1: Question types of WikiMovies dataset extracted from [32]

Question Type	**Example**
Movie to Director	who directed the movie Beneath the Darkness?
Movie to Writer	who is the writer of Anna Karenina?
Movie to Actors	who stars in The Deceivers?
Movie to Language	what is the language spoken in Eye of the Needle?
Movie to Year	what was the release year of the movie In Bloom?
Movie to Genre	what is the genre of the movie Holy Matrimony?
Movie to Tag	what topics is Eraser about?
Writer to Movie	which film did David Carter write the story for?
Director to Movie	what is a film directed by Eddie Murphy?
Actor to Movie	what films does William Atherton appear in?
Tag to Movie	what movies are about feminism?

6.2.2 MetaQA

MetaQA-2-hop and MetaQA-3-hop were introduced in 2018 as extensions to WikiMovies. They consist of more complicated questions: questions requiring 2-hop and 3-hop KG-traversal, respectively. Due to their multi-hop complexity, MetaQA datasets are more suitable for evaluating KGs, compared to WikiMovies. MetaQA-2-hop consists of 21 question types: 18 types in format of {*Actor*/*Writer*/*Director* **to** *Movie* **to** *Actor*/*Writer*/*Director* / *Year*/*Language*/*Genre*} plus 3 types in format of {*Movie* **to** *Actor*/*Writer*/*Director* **to** *Movie*}. Some examples are shown in Table 6.2. MetaQA-3-hop consist of 15 question types in format of {*Movie* **to** *Actor*/*Writer*/*Director* **to** *Movie* **to** *Actor*/*Writer*/*Director* / *Year*/*Language*/*Genre*}. Some examples are represented in Table 6.3. MetaQA-2-hop and MetaQA-3-hop include 14872 and 14274 questions in testing partition, respectively.

Multi-hop QA datasets are more suitable for evaluating KGs compared to 1-hop datasets. Back to the example presented in Introduction, to answer the 2-hop question of *In which movies did the director of Illuminata act?* the KG needs to identify *Illuminta*, *John Turturro*, and *Comany Man* as nodes; capture the *Director* relation between *Illuminta* and *John Turturro*; and also capture the *Actor* relation between *John Turturro* and *Comany Man* as demonstrated in Figure 1.1(a). Thus, a single 2-hop question verifies the veracity of several nodes and edges.

Table 6.2: Some questions of MetaQA-2-hop dataset captured from [60]

Question Type	**Example**
Director to Movie to Actor	who acted in the films directed by Gerard Johnson?
Writer to Movie to Language	what are the primary languages in the films written by Kevin Noland?
Actor to Movie to Genre	what are the genres of the movies acted by Odette Annable?
Movie to Actor to Movie	the actor of Black Swan also starred in which films?
Director to Movie to Year	when were the films directed by Peter Lord released?

Table 6.3: Some questions of MetaQA-3-hop dataset captured from [60]

Question Type	**Example**
Movie to Director to Movie to Actor	who acted in the films directed by the director of Curious George?
Movie to Writer to Movie to Genre	what types are the movies written by the writer of Lamerica?
Movie to Actor to Movie to Year	what were the release years of the films that share actors with the film Deep Red?
Movie to Writer to Movie to Director	who directed films for the writer of Swann in Love?

6.2.3 MDQA

Although MetaQA questions require multi-hop reasoning, they are mostly answerable with the content of a single document. In the case of MetaQA-2-hop, *Movie* is the middle node in 18 out of 21 question types. Since the documents are about movies, such questions are answerable with the content of a single document. For example, the question *What types are the movies directed by John Turturro?* is answerable with the content of the document describing *Illuminata*. In this document, it is mentioned that *Illuminata* is *comedy* and directed by *John Turturro*. Thus, by solely processing this document, we can report *comedy* as the answer. In the case of MetaQA-3-hop, *Movie* is the first and third node in all the question types. Thus, in most of the questions, the answer is available in the document of the given movie. For example, the 3-hop question of *Who acted in the films directed by the director of Illuminata?* can be simply answered by mentioning one of the actors of *Illuminata*.

The fact that the MetaQA questions are mostly answerable with a single document makes them less accurate in evaluating the quality of underlying KGs. In our experiments, many single-document questions are correctly answered despite the low quality of the KG in the corresponding subgraph. For example, consider the 2-hop question of *"Who directed the films written by Brandon Cole?"* which is answerable with the content of *Illuminata* article: *"Illuminata is a 1998 romantic comedy film directed by John Turturro and written by Brandon Cole and John Turturro"*. In an ideal KG, the answer path starts from *Brandon Cole*, passes through *Illuminata*, and ends in *John Turturro*. However, a faulty KG-generation framework may put an edge between *Brandon Cole* and *John Turturro* labeled with the whole sentence. Such KG is also capable of answering the given question despite being dirty. In conclusion, dense and dirty KGs can perform well in answering MetaQA questions.

On the other hand, there is no way to cheat in answering multi-document questions. The information must be correctly extracted from source documents and correctly aggregated into the KG so that the KG is capable of answering a multi-document question. Motivated by this observation, we propose **MDQA**, Multi-Document Question Answering dataset. MDQA consists of 2-hop questions extracted from the same Wikipedia corpus. The middle node in none of the questions is *Movie*. Thus, the questions are not answerable with a single document. MDQA is created by using the gold-standard KB proposed in [32].

MDQA consists of 9 question types represented in Table 6.4. Note that the first three question types are the same as the three multi-document question types in MetaQA-2-hop. The testing partition includes 10,000 questions. We suggest future work to also evaluate their framework on MDQA.

Table 6.4: Question types of MDQA dataset

Question Type	**count**	**Example**
Movie to Director to Movie	11412	Which movies have the same director as Lucky Jordan?
Movie to Writer to Movie	8817	The scriptwriter of Top Hat also wrote which movies?
Movie to Actor to Movie	11709	Which movies have the same actor as The Package?
Movie to Director-Actor to Movie	2017	In which movies the director of Reel Injun performed as an actor?
Movie to Director-Writer to Movie	7977	Which movies are written by the director of Midnight?
Movie to Writer-Director to Movie	6101	Which movies are directed by the scriptwriter of Gentlemen Broncos?
Movie to Writer-Actor to Movie	1839	The writer of Traitor played in which movies?
Movie to Actor-Director to Movie	4031	What are the movies directed by the actor of Perfect Sense?
Movie to Actor-Writer to Movie	3474	Which movies are written by the actor of Beyond Outrage?

6.3 Evaluation Metrics

Following the previous work, we use % hits@k to evaluate QA results [58]. Hits@k measures the ratio of questions for which the system retrieved a correct answer among the top k predictions. The authors of AutoKG reported % hits@1, % hits@3, and % hits@5 of their model on each dataset. During our experiments, specifically in the case of WikiMovies, we noticed that the ground truth is incomplete. For some questions, there are correct answers which are missing in the ground truth. Manually evaluating a sample of 20 Q&A

pairs from WikiMovies showed that in 5 out of 20 questions, the top answer retrieved by KG-Pipeline is correct but is missing in the ground truth. Due to such a high error rate, we decided to only report % hits@5, which is more reliable than % hits@1 and % hits@3. Clearly, the % hit@k evaluation error is less for higher values of k since there is a higher chance that at least one of the top k retrieved answers are included in the ground truth.

To compare our results with other QA systems, we report % hits@5 on WikiMovies, MetaQA-2-hop, MetaQA-3-hop, and MDQA. For each question, we check if any of the 5 retrieved answers (either in lower-case, title-case or upper-case form) is present in the ground truth. We refer to this evaluation approach as *exact answer evaluation*. The IR baseline discussed in Section 6.5.2 is evaluated in the same way.

AutoKG ends up reporting the best candidate paths, without further attempt for answer extraction. For calculating % hit@k scores, the authors counted a path as a correct path if the text form of the path contains one of the ground-truth answers as a substring. We refer to this approach as *path evaluation*. Clearly, higher scores are reported with path evaluation compared to exact answer evaluation since the answer extraction error is not reflected in the former approach. We also calculated our % hits@5 results with path evaluation approach to compare our results to AutoKG's in a fair way.

6.4 System Configuration

We implemented our pipeline on top of Spacy pipeline [19]. We used Python 3.7[1] for all implementations. Allennlp models (for OpenIE, coreference resolution, and NER), roberta-base-squad2 (QA model), and all-MiniLM-L6-v2 (sentence similarity model) are all used with default parameters, without any fine-tuning. MUSS is used with the following hyper-parameters:

```
LengthRatioPreprocessor = 1
ReplaceOnlyLevenshteinPreprocessor = 0.8
WordRankRatioPreprocessor = 1
DependencyTreeDepthRatioPreprocessor = 0.4
```

`hops` and `beam_size` are the main hyper-parameters of the QA algorithm. To achieve the best results, we set `hops` to one more than the hops of the input dataset. For example, for MetaQA-2-hop and MDQA, we run the algorithm with `hops = 3`. We used

[1]MUSS, the model used in the sentence simplification component, is not compatible with recent Python versions like Python 3.9.

`beam_size = 10` in all experiments. We did not use the training partition of the datasets since our KG-based QA approach does not incorporate any trainable components. Development partitions are used for hyper-parameter tuning.

We conducted our experiments on a Linux Ubuntu 21.10 server with 24 CPU cores and an NVIDIA 2080 Ti GPU. The GPU makes the simplification step much faster. However, the whole pipeline can be executed without GPU. Note that the pipeline has a high usage of memory. The size of the MUSS is 4.8GB and the total size of pipeline models is about 2GB. Thus, at least 8GB of free memory is recommended for running the pipeline. More implementation details will be published along with code in our Github repository.

6.5 Question Answering Results

In this section, we present the QA results in two scenarios. In the first scenario, we compare our results to AutoKG, the only previous work on QA with automatically-generated KGs. The results demonstrate the advantage of our pipeline in answering complicated questions, showing the higher quality of our KGs. In the second scenario, we compare our QA results to an IR baseline. We show that in addition to explainability and capability of answering multi-document questions, our pipeline performs well in terms of %hits@5 QA results, compared to the IR baseline.

In the following results, we incorporated a row specified with *Fine-tuned KG-Pipeline*, identifying the results of our pipeline when fine-tuned on the movie domain. In Section 4.3, we claimed that KG-Pipeline is a domain-agnostic framework that can be fine-tuned on various domains. In this fine-tuned version, a slight modification is made: We intervene in the NER component so that it recognizes production years, movie genres, and languages as entities. Since the QA datasets include year, genre, and language questions, we expect an improvement in the QA results when such attributes are recognized as KG nodes.

6.5.1 Comparison to AutoKG

Hits@5 results of our model are compared to AutoKG in Table 6.5. As discussed in Section 6.3, we used the path evaluation approach to calculate the result of KG-Pipeline in Table 6.5. While KG-Pipeline performs close to AutoKG in WikiMovies and MetaQA-2-hop, it outperforms AutoKG in MetaQA-3-hop by a large margin. We believe this is due to the higher quality of our KGs, as our KG-traversal algorithm is analogous to the algorithm

Table 6.5: %hits@5 QA results calculated with path evaluation approach

Model	WikiMovies	MetaQA-2hop	MetaQA-3hop
AutoKG	64.48	**47.04**	32.86
KG-Pipeline	63.16	44.11	44.66
Fine-tuned KG-Pipeline	**67.36**	42.51	**45.44**

described in the AutoKG paper. The source code of AutoKG is not published so we could not compare the models on the proposed MDQA dataset.

6.5.2 Comparison to IR baseline

We discussed that KG-based QA has two advantages compared to the traditional IR-based QA:

- Unlike IR-based QA systems, KG-based QA systems are explainable.
- Unlike IR-based QA systems, KG-based QA systems are capable of combining information extracted from various sources.

In this section, we aim to show that our KG-based QA system also performs as well as IR-based systems in terms of QA results. To this end, we compare the QA results of KG-Pipeline to *BM25+RoBERTa*, as an IR-based baseline. In this baseline, we first utilize BM25 to search the corpus for the most relevant documents to the input question. Then, we apply roberta-base-squad2 [29], the same model as the one used in our QA framework, to extract the answer from documents retrieved by BM25. In our experiments, we used the Pyserini [28] implementation of BM25. Note that our KG-Pipeline does not use the training partitions of datasets. Thus, we avoided fine-tuning the RoBERTa model on training datasets to conduct a fair comparison.

The QA results are shown in Table 6.6. As discussed in Section 6.3, we used the exact answer evaluation approach to calculate the scores in Table 6.6. According to the results, KG-Pipeline outperforms the baseline in 3 out of 4 datasets. Since the same answer extraction model (roberta-base-squad2) is used in both systems, we can claim that our KG-traversal approach is a more effective way of retrieving relevant information to a candidate query, compared to BM25. The results of models on the MDQA dataset verify our claim

Table 6.6: %hits@5 QA results calculated with exact answer evaluation approach

Model	**WikiMovies**	**MetaQA-2hop**	**MetaQA-3hop**	**MDQA**
BM25+RoBERTa	57.65	**43.66**	35.12	2.60
KG-Pipeline	56.88	36.16	35.15	25.88
Fine-tuned KG-Pipeline	**64.70**	38.21	**41.58**	**25.88***

$*$ Since MDQA does not include any genre, language, or year questions, the fine-tuned version would not be helpful in the case of MDQA. Thus, we repeated the result of the base version.

about the capability of KG-Pipeline in answering some multi-document questions while the IR-based approach is incapable of answering such questions.

Chapter 7

Conclusion and Future Work

We formulated a solution framework to the problem of automatic KG generation. The framework constitutes of well-known NLP components, namely sentence simplification, open information extraction, coreference resolution, and named entity recognition. The NLP tasks are separately applied to the input texts. The outputs are combined and processed within a graph generation module, resulting in a KG. This modular design, unlike the previous work on KG generation, is not dependent on any specific implementations for information extraction or other NLP tasks. Thus, with the advent of superior NLP models in future, the pipeline can be upgraded by replacing the implementation of each component with state-of-the-art models. To showcase the advantage of our pipeline, we focused on question answering as a downstream application of KG creation. Our KG generation pipeline, combined with a simple KG-traversal algorithm, performs as well as baselines in answering simple factoid questions while outperforming them in answering more complex, specifically multi-document, questions.

There are several directions for future work to consolidate the current project. The notable ones are as follows:

- Our system is originally designed for capturing binary relationships between entities. The current framework is incapable of capturing more complicated forms of relations in which more than two entities are associated. Treating such relationships as binary relations may lead to augmenting KGs with noisy information. For example, taking *"Person[A] believes that Movie[B] is directed by Person[C]"* as input, the system simplistically infers *director* relation between the nodes representing *Movie[B]* and *Person[C]* although this might not be a correct fact. Improving the system to capture more advanced relations and distinguish between facts and opinions is one important direction for future work.

- We discussed that we used state-of-the-art models for sentence simplification, coreference resolution, and open information extraction. However, an ablation analysis for each component is required to be conducted to formally prove the superiority of the current models in the KG generation task. Specifically, since we introduced sentence simplification as a useful preprocessing step, an ablation analysis is needed to show the effect of sentence simplification component.

- We noticed that splitting complex sentences with MUSS noticeably improves the quality of extracted triples and the resulting KG. It is the first usage of a model from sentence simplification literature in OpenIE. In future, experiments can be conducted to formally demonstrate the effect of extending RNNOIE by adding MUSS in OpenIE literature. In other words, experiments on the OpenIE benchmarks can be conducted to locate the position of RNNOIE + MUSS among other state-of-the-art models shown in Figure 3.1.

- As discussed in Section 3.3, we used a simple QA algorithm similar to AutoKG's. State-of-the-art KBQA models are designed for high-quality manually-annotated KGs. At first glance, such models do not seem to be applicable to our automatically generated KGs. However, an experimental study is required to test popular KBQA algorithms on the generated KGs.

- As discussed in Section 6.1, the literature lacks a standard way of directly evaluating automatically-generated KGs. Specifically, since we do not apply ontology mapping, the extracted predicates can be in various forms. Thus, it is unclear how to calculate the precision/recall of the extracted triples against a gold standard KB. Consequently, we preferred to evaluate KGs indirectly, i.e., in QA as a downstream application of KGs. In future, an attempt is required to find out a robust way of directly evaluating KGs.

- As discussed in Section 4.3, predicate mapping standardizes the generated KGs while reducing their coverage and QA recall. More experiments can be conducted in future to test the pipeline with a predicate mapping component, and measure the degree to which predicate mapping harms the QA results.

- We conducted all the experiments on the document corpus of WikiMovies. Experiments on other QA datasets can be conducted in future to further evaluate the quality of our pipeline in different domains. Moreover, we only used BM25 + RoBERTa as the IR baseline to compare with our QA framework; experiments with more advanced baselines should be added in future versions.

- Finally, our generated KGs are still far from the human-annotated KGs, specifically in terms of the quality of extracted edge labels. On many input questions, our QA module fails to retrieve the correct answer even though the answer is available in the neighbourhood of the seed node. In future versions of KG-Pipeline, we aim to improve the system by focusing on predicate extraction as well as the KG-traversal algorithm.

References

[1] Allennlp's coreference resolution demo. https://demo.allennlp.org/coreference-resolution. Accessed: 2021-06-07.

[2] Allennlp's named entity recognition demo. https://demo.allennlp.org/named-entity-recognition/named-entity-recognition. Accessed: 2021-06-07.

[3] Allennlp's open information extraction demo. https://demo.allennlp.org/open-information-extraction. Accessed: 2021-06-07.

[4] Fernando Alva-Manchego, Carolina Scarton, and Lucia Specia. Data-driven sentence simplification: Survey and benchmark. *Computational Linguistics*, 46(1):135–187, 2020.

[5] Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, 2015.

[6] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[7] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, page 2670–2676, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[8] Bojan Bozic, Jayadeep Kumar Sasikumar, and Tamara Matthews. Knowtext: Auto-generated knowledge graphs for custom domain applications. In *The 23rd Interna-*

tional Conference on Information Integration and Web Intelligence, pages 350–358, 2021.

[9] Ermei Cao, Difeng Wang, Jiacheng Huang, and Wei Hu. Open knowledge enrichment for long-tail entities. In *Proceedings of The Web Conference 2020*, pages 384–394, 2020.

[10] Matthias Cetto, Christina Niklaus, André Freitas, and Siegfried Handschuh. Graphene: Semantically-linked propositions in open information extraction. *arXiv preprint arXiv:1807.11276*, 2018.

[11] Janara Christensen, Stephen Soderland, and Oren Etzioni. An analysis of open information extraction based on semantic role labeling. In *Proceedings of the sixth international conference on Knowledge capture*, pages 113–120, 2011.

[12] Janara Christensen, Stephen Soderland, and Oren Etzioni. An analysis of open information extraction based on semantic role labeling. In *Proceedings of the sixth international conference on Knowledge capture*, pages 113–120, 2011.

[13] Lei Cui, Furu Wei, and Ming Zhou. Neural open information extraction. *arXiv preprint arXiv:1805.04270*, 2018.

[14] Luciano Del Corro and Rainer Gemulla. Clausie: clause-based open information extraction. In *Proceedings of the 22nd international conference on World Wide Web*, pages 355–366, 2013.

[15] Eleftherios Dimitrakis, Konstantinos Sgontzos, and Yannis Tzitzikas. A survey on question answering systems over linked data and documents. *Journal of intelligent information systems*, 55(2):233–259, 2020.

[16] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 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*, pages 601–610, 2014.

[17] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 1535–1545, 2011.

[18] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. Allennlp: A deep semantic natural language processing platform. 2017.

[19] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spacy: Industrial-strength natural language processing in python. 2020.

[20] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[21] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

[22] Mandar Joshi, Omer Levy, Daniel S Weld, and Luke Zettlemoyer. Bert for coreference resolution: Baselines and analysis. *arXiv preprint arXiv:1908.09091*, 2019.

[23] Natthawut Kertkeidkachorn and Ryutaro Ichise. An automatic knowledge graph creation framework from natural language text. *IEICE TRANSACTIONS on Information and Systems*, 101(1):90–98, 2018.

[24] Keshav Kolluru, Vaibhav Adlakha, Samarth Aggarwal, Soumen Chakrabarti, et al. Openie6: Iterative grid labeling and coordination analysis for open information extraction. *arXiv preprint arXiv:2010.03147*, 2020.

[25] Keshav Kolluru, Samarth Aggarwal, Vipul Rathore, Soumen Chakrabarti, et al. Imojie: Iterative memory-based joint open information extraction. *arXiv preprint arXiv:2005.08178*, 2020.

[26] Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the fifteenth conference on computational natural language learning: Shared task*, pages 28–34, 2011.

[27] Kenton Lee, Luheng He, and Luke Zettlemoyer. Higher-order coreference resolution with coarse-to-fine inference. *arXiv preprint arXiv:1804.05392*, 2018.

[28] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th Annual*

International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021), pages 2356–2362, 2021.

[29] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[30] Louis Martin, Angela Fan, Éric de la Clergerie, Antoine Bordes, and Benoît Sagot. Muss: multilingual unsupervised sentence simplification by mining paraphrases. *arXiv preprint arXiv:2005.00352*, 2020.

[31] Louis Martin, Benoît Sagot, Eric de la Clergerie, and Antoine Bordes. Controllable sentence simplification. *arXiv preprint arXiv:1910.02677*, 2019.

[32] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*, 2016.

[33] Amit Mishra and Sanjay Kumar Jain. A survey on question answering systems with classification. *Journal of King Saud University-Computer and Information Sciences*, 28(3):345–361, 2016.

[34] Spacy neuralcoref library. https://spacy.io/universe/project/neuralcoref. Accessed: 2021-07-07.

[35] Christina Niklaus, Bernhard Bermeitinger, Siegfried Handschuh, and André Freitas. A sentence simplification system for improving relation extraction. *arXiv preprint arXiv:1703.09013*, 2017.

[36] Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. A survey on open information extraction. *arXiv preprint arXiv:1806.05599*, 2018.

[37] Harinder Pal et al. Demonyms and compound relational nouns in nominal open ie. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 35–39, 2016.

[38] Harinder Pal et al. Demonyms and compound relational nouns in nominal open ie. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 35–39, 2016.

[39] Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*, 2017.

[40] Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher D Manning. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 492–501, 2010.

[41] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[42] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[43] Arpita Roy, Youngja Park, Taesung Lee, and Shimei Pan. Supervising unsupervised open information extraction models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 728–737, 2019.

[44] Swarnadeep Saha et al. Open information extraction from conjunctive sentences. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2288–2299, 2018.

[45] Swarnadeep Saha, Harinder Pal, et al. Bootstrapping for numerical open ie. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 317–323, 2017.

[46] Michael Schmitz, Stephen Soderland, Robert Bart, Oren Etzioni, et al. Open language learning for information extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 523–534, 2012.

[47] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[48] Gabriel Stanovsky, Jessica Ficler, Ido Dagan, and Yoav Goldberg. Getting more out of syntax with props. *arXiv preprint arXiv:1603.01648*, 2016.

[49] Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. Supervised open information extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 885–895, 2018.

[50] Nikolaos Stylianou and Ioannis Vlahavas. A neural entity coreference resolution review. *Expert Systems with Applications*, 168:114466, 2021.

[51] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[52] Gerhard Weikum, Xin Luna Dong, Simon Razniewski, Fabian Suchanek, et al. Machine knowledge: Creation and curation of comprehensive knowledge bases. *Foundations and Trends® in Databases*, 10(2-4):108–490, 2021.

[53] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[54] Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. Corefqa: Coreference resolution as query-based span prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6953–6963, 2020.

[55] Xindong Wu, Jia Wu, Xiaoyi Fu, Jiachen Li, Peng Zhou, and Xu Jiang. Automatic knowledge graph construction: A report on the 2019 icdm/icbk contest. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 1540–1545. IEEE, 2019.

[56] Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415, 2016.

[57] Vikas Yadav and Steven Bethard. A survey on recent advances in named entity recognition from deep learning models. *arXiv preprint arXiv:1910.11470*, 2019.

[58] Seunghak Yu, Tianxing He, and James Glass. Autokg: Constructing virtual knowledge graphs from unstructured documents for question answering. *arXiv preprint arXiv:2008.08995*, 2020.

[59] Junlang Zhan and Hai Zhao. Span model for open information extraction on accurate corpus. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9523–9530, 2020.

[60] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. Variational reasoning for question answering with knowledge graph. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[61] Chen Zhao, Chenyan Xiong, Xin Qian, and Jordan Boyd-Graber. Complex factoid question answering with a free-text knowledge graph. In *Proceedings of The Web Conference 2020*, pages 1205–1216, 2020.

[62] Xiaohan Zou. A survey on application of knowledge graph. In *Journal of Physics: Conference Series*, volume 1487, page 012016. IOP Publishing, 2020.