K^2-GNN: Multiple Users' Comments Integration with Probabilistic K-Hop Knowledge Graph Neural Networks

Huixin Zhan HUIXIN.ZHAN@TTU.EDU
Department of Computer Science, Texas Tech University, Lubbock, TX 79409, USA

Kun Zhang KZHANG@XULA.EDU
Department of Computer Science, Xavier University of Louisiana, New Orleans, LA 70125, USA

Chenyi Hu CHU@UCA.EDU
Department of Computer Science, University of Central Arkansas, Conway, AR 72035, USA

Victor S. Sheng* VICTOR.SHENG@TTU.EDU
Department of Computer Science, Texas Tech University, Lubbock, TX 79409, USA

***Corresponding author: Victor S. Sheng, victor.sheng@ttu.edu**

Editors: Vineeth N Balasubramanian and Ivor Tsang

Abstract

Integrating multiple comments into a concise statement for any online products or web services requires a non-trivial understanding of the input. Recently, graph neural networks (GNN) has been successfully applied to learn from highly-structured graph representations to mitigate the relationship between entities, such as co-references. However, current inter-sentence relation extraction cannot leverage discrete reasoning chains over multiple comments. To address this issue, in this paper, we propose a probabilistic K-hop knowledge graph (KKG) to extend existing knowledge graphs with inferred relations via discrete intra-sentence and inter-sentence reasoning chains. KKG associates each inferred relation with a confidence value through Bayesian inference. We further answer how a knowledge graph with inferred relations can help the multiple comments integration through integrating KKG with GNN (K^2-GNN). Our extensive experimental results show that our K^2-GNN outperforms all baseline graph models on multiple comments integration.

Keywords: Graph neural networks, Knowledge graph, Web data integration, Graph representation

1. Introduction

Our daily lives are increasingly dependent on information. However, a great amount of information is generated every day, so that manual processing is impossible. For example, there are 759 global ratings and 281 global reviews for the book entitled "Data Science for Business" on Amazon[1]. For decades, research in natural language processing (NLP) has focused on automatic summarization for this problem. Most efforts have focused on single-document summarization (Zajic et al., 2008). Recently, a few multi-document summarization systems proposed to extract relevant or interesting portions from a set of documents (Radev et al.,

1. https://www.amazon.com/Data-Science-Business-Data-Analytic&-Thinking/dp/1449361323/ref=sr_1_1?crid=2RFHEYPG74LUT&dchild=1&keywords=data+science+for+business&qid=1611113995&sprefix=data+science+for+%2Caps%2C182&sr=8-1

2004; Erkan and Radev, 2004; Gao et al., 2019). Multi-document summarization is more complex than single-document summarization (Zajic et al., 2008). One of difficulties arises from the thematic diversity among a large set of documents.

Multiple comments integration is even more difficult than existing multi-document summarization, since (1) the comments of products or services are not only noisy but also could contain redundant and conflicting information among the themes, and (2) the comments are informal. According to our knowledge, there are a few works related to crowdsourcing-based summarization, e.g., (Rong et al., 2020; Jiang et al., 2018; Gao et al., 2019). For example, Rong et al. (2020) proposed a self-play DQN for multiple comments integration. Jiang et al. (2018) proposed solving query-based summarization via human annotators, and Gao et al. (2019) proposed a sequence to sequence framework with attention for reader-aware summarization. However, there is no existing graph-based work leveraging the benefits of graphs to address crowdsourced data summarization. Some recent results have shown the promising of applying GNN on graphs (Fernandes et al., 2018).

KG presents facts in graphs, consisting of entities, relations, and semantic descriptions, where the relations represent the relationships between entities. Usually, entities relations contain types and properties with a well-defined meaning, but they fail to model complex relation paths (Ji et al., 2020). Inter-sentence relation extraction deals with a number of complex semantic relationships in documents, e.g., local, non-local, syntactic and semantic dependencies (Sahu et al., 2019). However, reasoning discrete inter-sentence logic-wise reasoning chains on weakly structured data such as documents remains a challenge. In order to deal with a number of complex logic relations in documents, subject and object dependencies need to be first extracted. Until now, there are several related multi-hop reasoning methods exploring such dependencies in knowledge graph (KG) (Lao et al., 2011; Xiong et al., 2017), **but it's not clear that how multi-hop reasoning will contribute to summarization**. Most recent research focuses on injecting logical rules to improve reasoning, with joint learning or iterative training by incorporating first-order logic rules (Zhang et al., 2019). However, to the best of our knowledge, (1) no existing works explore how reasoning through multi-hop nodes will help summarization; and (2) no existing works iteratively build high-quality KKG with reasoning confidences by incorporating Bayesian inference.

In this paper, we propose a novel KKG for multiple users' comments to construct a multiple comments level graph. The nodes of the graph correspond to subjects and objects and the edges of the graph represent local and non-local dependencies between subjects and objects. Specifically, the local dependencies represent intra-sentence relations, which can be constructed through building clauses; the non-local dependencies represent inter-sentence relations, which can be iteratively learned through Bayesian inference. Bayesian inference will provide a confidence for each "correctly firing" output relation. In each iteration, after an expert inspects and verifies the output relation with the highest confidence, Bayesian inference will recompute the confidences of the remaining relations according to the expert's feedback. The contributions of this paper are threefold. Firstly, we propose a novel model KKG for inter-sentence relation extraction, so that we can simultaneously capture local and non-local dependencies in multi-hop reasoning. Secondly, we propose a systematic methodology to infer the confidence of each edge in an initial derivation graph using Bayesian networks. Finally, we develop a novel graph-based K^2-GNN approach to perform multiple

comments integration with possible conflicts (inconsistency) and redundancy in the weakly structured comments.

2. Related Works

In this section, we summarize related works from three aspects, inter-sentence relation extraction, multiple users' comments integration, and multi-hop reasoning.

2.1. Inter-sentence Relation Extraction

In order to extract inter-sentence relations, most approaches utilize distant supervision to automatically generate document-level corpora (Song et al., 2018). Recently, Verga et al. (2018) introduced the multi-instance learning (Riedel et al., 2010) to treat multiple mentions of target entities in a document. Inter-sentence relations depend not only on local but also on non-local dependencies. Dependency trees are often used to extract local dependencies of semantic relations (Liu et al., 2015) in intra-sentence relation extraction. However, such dependencies are not adequate for inter-sentence relation extraction, since different sentences have different dependency trees. To capture their relations, it is essential to connect co-referring entities (e.g., Oxytocin and Oxt, RNNs and CNNs). Connecting co-references are often used for intra-sentence relation extraction (Lin et al., 2016), but it is not effective on longer sequences (Sahu and Anand, 2018), because it fails to capture non-local dependencies. One of the most recent work combines local and non-local dependencies, and builds a labelled edge Graph CNN (GCNN) model (Marcheggiani and Titov, 2017) on a document-level graph. The document-level graph is formed by connecting words with local dependencies from syntactic parsing and sequential information, as well as non-local dependencies from co-reference resolution and other semantic dependencies (Peng et al., 2017). However, all the aforementioned approaches are corresponding to non-logic inter-sentence relation extraction, and no existing methods address inter-sentence logic relation through reasoning.

2.2. Graph-based Users' Comments Integration

In a crowdsourcing scenario, individuals or organizations obtain multiple comments of online products or services from a large, relatively open and often rapidly evolving group of internet users (Zhang et al., 2016). In this paper, we aim at integrating multiple comments for any products or services, which are posted by participants (customers) with high inconsistency and redundancy. It is obvious that integrating such multiple comments together is a challenging problem. According to our knowledge, there are works focusing on summarizing the crowdsourced data (Rong et al., 2020; Jiang et al., 2018; Gao et al., 2019). However, there are currently no existing GCN works addressing this problem.

3. Preliminaries

We start by defining our notation, and then we present our proposed KKG.

S_1: muhammadu buhari tells S_2: cnn's christiane amanpour that he will fight S_3: corruption in S_4: nigeria. Around S_5: 9 percent of Nigerians considered corruption to be the most important problem facing their country, a significant decrease from the S_6: 14 percent recorded in the 2016 survey. Nigeria is the most populous country in S_7: africa and is grappling with violent bokoharam S_8: extremists.

Figure 1: A natural language segment example

3.1. Notations

Let $\mathcal{G} = (V, E)$ be an undirected graph consisting of a set of nodes V and a set of edges E. The neighborhood of radius k (or k-hop neighborhood) of a node v is the set of nodes at a distance less than or equal to k from v, and denoted as $\mathcal{N}_k(v)$. For an edge $vv_1 \in E$, edge labelling is a function $\phi(vv_1)$ to map an edge to a binary value (e.g., $\{0, 1\}$) in typical KG. However, in our proposed KKG, $\phi(vv_1)$ is labeled by the probability P_r given by Bayesian inference. A *virtual edge* $\widehat{vv_1}$ is defined as an edge that connects the node v with its k-hop neighborhood (when $k > 1$) $v_1 \in \mathcal{N}_k(v)$ with the probability P_r given by Bayesian inference.

3.2. Motivation Examples

Considering a KG in Figure 2, obtained from the natural language segment from the CNN/DM dataset (Chen et al., 2016) shown in Figure 1. In typical KG, all nodes are connected to their 1-hop neighborhood with $\phi(vv_1) = 1, \forall v, v_1 \in \mathcal{N}_k(v)$. Direct intra-sentence logic relations from subjects to objects are presented as triples[2], e.g., $S_3 \xrightarrow{1} S_5$.

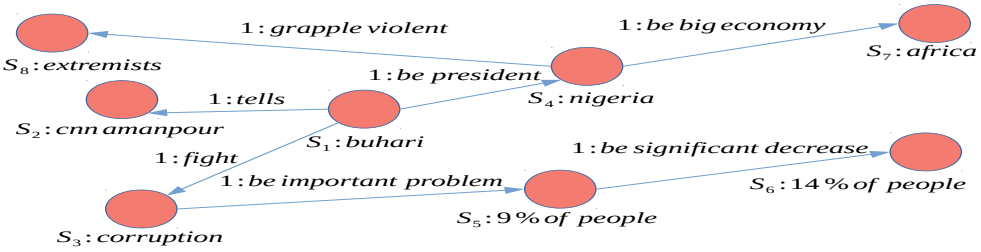

Figure 2: A KG that makes use of triples {a subject,a predicate and an object}, where **subjects** and **objects** are modeled as nodes, and the **predicates** are modeled as edges pointing from subjects to objects. Direct intra-sentence logic relations are labeled as 1.

However, in constructing the KKG, when we consider the relations of each node with its K-hop neighborhood, if there exists a direct logic relation between a pair of nodes, the direct logic relation can be reasoned by an expert. However, there are no edges between the

2. A triple is a set that is composed of {a subject, a predicate, an object} in a typical KG

node and its K-hop neighbors when $K > 1$. Since existing knowledge does not fully extract the inter-sentence logic relations, we propose to add virtual edges between the node and its K-hop neighbors with logic reasoning in our proposed KKG, where these virtual edges are labeled with the confidence of the corresponding "correctly firing" inference. For example, in Figure 2, S_3 has a direct logic relation with S_6 that can be reasoned by learning explicit inference formulas from an expert. For example, if the KG includes the beliefs such as

$$\begin{aligned} \{S_5S_6\} \triangleq\ & S_5 : \underline{\textbf{9\% of people}} \xrightarrow{P_r(S_5S_6)} S_6 : \textbf{14\% of people} \\ \{S_3S_5\} \triangleq\ & S_3 : \textbf{corruption} \xrightarrow{P_r(S_3S_5)} S_5 : \underline{\textbf{9\% of people}}. \end{aligned} \tag{1}$$

The inference from S_3 to S_6 can be shown as the following connective relation:

$$\begin{aligned} &\{S_5S_6\} \wedge \{S_3S_5\} \xrightarrow{P_r(\widehat{S_3S_6})} \textbf{corruption is an important} \\ &\textbf{problem considered by 14\% of people (in 2016)}. \end{aligned} \tag{2}$$

Note that these three clauses[3] are not always tautologies because of they do not include the complete set of correct reasoned objects. Therefore, the valuation of a clause is true with probability P_r in our KKG, where a valuation is a function to assign each node S_i a unique boolean value. For example, the reasoned object "corruption is an important problem considered by 14% of people (in 2016)" is with a probability $P_r(\widehat{S_3S_6})$. We can also notice that the subject of the former clause $\{S_5S_6\}$ is just the object of the latter clause $\{S_3S_5\}$. Therefore, we can identify all the possible logic relations following this *subject-object flow*. Now, we can conclude the central question of this paper is how we can learn the probability of the inference "correctly firing" of inter-sentence logic relations through the *subject-object flow* and how the KKG contribute to summarization.

4. The Proposed KKG

In this section, we introduce the interaction model for constructing the KKG. In subsection 4.1, to make the explanation simple, we first fix the rule r with a firing probability $P_r = 0.95$ in each clause to illustrate how our proposed interaction model guides the valuation of the inference away from false positives and towards output relations that have a high probability to be true, and then iteratively constructs the high-quality KKG. In subsection 4.2, we introduce how to learn the rule firing probability P_r via Bayesian networks.

4.1. Construction of the KKG

In this subsection, we first discuss how to build a non-weighted KKG $\mathcal{G}$ by applying these logic relation analysis rules in a typical KG, and then introduce how to compute the weights of $\mathcal{G}$ under the fixed P_r.

Build a non-weighted KKG $\mathcal{G}$ Before discussing how to build a non-weighted KKG, we need to first define the input hypothesis, the output relation, and the analysis rules for logic relation analysis, As shown in Table 1, this analysis takes the hypothesis $H(S_i, S_j)$

3. A clause is a group of words that contains a triple

Table 1: The logic relation analysis in a typical KG.

Input hypothesis $H(S_i, S_j) \leftarrow$ the valuation of the clause between S_i and S_j is true
Output relation $c(S_i, S_j) \leftarrow$ the valuation of the inference between node S_i and S_j is true
Analysis rules $r(S_i, S_j) \leftarrow (c(S_i, S_j), H(S_j, S_m) \rightarrow c(S_i, S_m))$ when S_i is the k-hop neighborhood of S_j with $k = 1$ to K (See subsection 4.2.1).

as the input, applies the analysis rule $r(S_i, S_j)$, and produces the relation $c(S_i, S_j)$ as the output. In all relations, variable S_i and S_j range over the domain of all nodes. The analysis rules are intended to be read from left-to-right with all variables universally quantified. For example, the analysis rule $r(S_i, S_j)$ means "For all nodes S_i, S_j, and S_m, if the valuation of the inference between S_i and S_j is true, and the valuation of the clause between S_j and S_m is true, then the valuation of the inference between node S_i and S_m is true."

With the definitions above, we can discuss how to build a non-weighted KKG $\mathcal{G}$ by applying the aforementioned logic relation analysis rules in a typical KG. To apply the analysis of Table 1 to the KG in Figure 2, one starts with the set of existing relations, and repeatedly applies the analysis rule until no new facts can be derived. Starting with the relation $c(S_1, S_1)$, we show a portion of the derivation graph obtained in Figure 3. Each box represents a tuple, and is shown in blue if it is an input hypothesis. Nodes identified with rule names represent *grounded clauses*. For example, the node $r(S_1, S_3)$ indicates the "grounded instance" of the rule r. This clause takes as a hypothesis (the tuple $H(S_1, S_3)$) and derives the output relation $c(S_1, S_3)$, and the arrows represent these dependencies. Observe that reasoning chains are conjunctive: a rule fires iff all of its antecedents are derivable. Note that for K-hop neighborhood pairs, this analysis rule is repeatedly applied for K times. That is, it is applied one for every different hop. Besides, the analysis is flow-sensitive. That is, it takes into account the order of the edges with direction, represented by the relation $H(S_i, S_j)$ and $c(S_i, S_j)$, but it is path-insensitive. That is, it disregards the satisfiability of path conditions and predicates along branches.

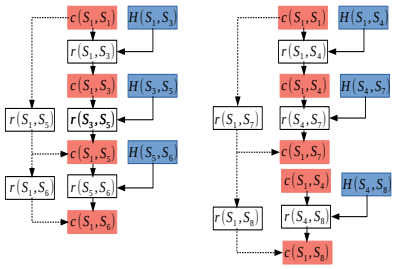

Figure 3: Portion of the derivation graph obtained by applying the static rule for logic relation analysis to the KG in Figure 2.

Assign the weights for $\mathcal{G}$ under a fixed P_r After we have built the non-weighted KKG $\mathcal{G}$, we are going to introduce how to assign the weights for $\mathcal{G}$ under a fixed P_r. As

we mentioned before, incomplete output relations are the principal cause of false reasoning. Since the number of the incomplete output relations could be very large, we save the expert's time and cost by selecting the output relation with the highest confidence. Specifically, we first assign the valuation of all output relations as true, and then compute the confidence for each output relation. The output relation with the highest confidence is then assigned to the expert to valuate. If the expert labels it as true, then all the weights that are computed from the prior belief hold; if an output relation c_f is labeled by the expert as false, the weights for each relation c need to be recomputed by replacing the prior belief $P_r(c)$ for each output relation c with the posterior belief $P_r(c|\neg r_f)$. Note that the valuation $\mathcal{V}(c_f) = F$ is equivalent to $\mathcal{V}(r_f) = F$ in our logic analysis. We consider a case that $c(S_1, S_3)$ and $H(S_3, S_5)$ are all true, while $c(S_1, S_5)$ is not always true. We address this problem by relaxing the interpretation of clause nodes, and only treat them probabilistically:

$$P_r(r(S_3, S_5)|c(S_1, S_3) \wedge H(S_3, S_5)) = 0.95, \textbf{ and} \tag{3}$$
$$P_r(\neg r(S_3, S_5)|c(S_1, S_3) \wedge H(S_3, S_5)) = 1 - 0.95 = 0.05, \tag{4}$$

where $P_r = 0.95$ is the probability of the clause "correctly firing", by setting it to a value that is less that 1. We make it possible for the output relation $c(S_1, S_5)$ to still be false. If any of the antecedents of $r(S_3, S_5)$ is false, then it is definitely false:

$$P_r(r(S_3, S_5)|\neg(c(S_1, S_3) \wedge H(S_3, S_5))) = 0, \textbf{ and} \tag{5}$$
$$P_r(\neg r(S_3, S_5)|\neg(c(S_1, S_3) \wedge H(S_3, S_5))) = 1. \tag{6}$$

Actually, these rule firing probabilities can be learning. How to learn these rule firing probabilities will be discussed in subsection 4.2.3. At this time, we just associate the rule r with the firing probability $P_r = 0.95$. To simplify the discussion, we treat $c(S_1, S_1)$ as an initial output relation with $P_r(c(S_1, S_1)) = 0.6$. By attaching conditional probability distributions (CPDs) such as Equations (3-6) to each node of Figure 3, we view the derivation graph as a Bayesian network. Specifically, marginal inferences are performed on the network to associate each output relation with a probability (or belief) of being a true inference. For example, it computes the probability of the output relation $c(S_1, S_5)$ as follows:

$$\begin{aligned} P_r(c(S_1, S_5)) &= P_r(c(S_1, S_5) \wedge r(S_3, S_5)) + P_r(c(S_1, S_5) \wedge \neg r(S_3, S_5)) \\ &= P_r(c(S_1, S_5) \wedge r(S_3, S_5)) \\ &= P_r(c(S_1, S_5)|r(S_3, S_5))P_r(r(S_3, S_5)) \\ &= P_r(r(S_3, S_5)|c(S_1, S_3) \wedge H(S_3, S_5)) \\ &\qquad \cdot P_r(c(S_1, S_3))Pr(H(S_3, S_5)) \\ &= 0.95\ P_r(c(S_1, S_3)) = 0.95^2 \cdot 0.6 = 0.5415 \end{aligned} \tag{7}$$

An expert now inspect this output relation $c(S_1, S_5)$. If the expert classifies $r(S_1, S_5)$ as a true relation, then $P_r(c(S_1, S_5)) = 0.5415$ holds. If $r(S_1, S_5)$ is classified by the expert as a false relation, then we need to replace $P_r(c)$ with the posterior belief $P_r(c|\neg r(S_1, S_5))$ for each output relation c. That is, we can effectively propagate the expert's feedback to the remaining relations. The computation of the updated confidence values occurs by a similar

procedure as before. For example:

$$\begin{aligned} P_r(c(S_1,S_6)|\neg r(S_1,S_5)) = \\ & P_r(c(S_1,S_6)\wedge c(S_1,S_5)|\neg r(S_1,S_5))+ \\ & P_r(c(S_1,S_6)\wedge \neg c(S_1,S_5)|\neg r(S_1,S_5)) \\ & = P_r(c(S_1,S_6)\wedge c(S_1,S_5)|\neg r(S_1,S_5)) \\ & = P_r(c(S_1,S_6)|c(S_1,S_5))P_r(c(S_1,S_5)|\neg r(S_1,S_5)) \\ & = 0.95^2 P_r(c(S_1,S_5)|\neg r(S_1,S_5)), \end{aligned} \tag{8}$$

finally, by Bayes' rule, we have:

$$\begin{aligned} & P_r(c(S_1,S_5)|\neg r(S_1,S_5)) \\ & = \frac{P_r(\neg r(S_1,S_5)|c(S_1,S_5))\cdot Pr(c(S_1,S_5))}{P_r(\neg r(S_1,S_5))} \\ & = \frac{0.05\cdot 0.95^2\cdot 0.6}{0.6} = 0.0451 \end{aligned} \tag{9}$$

Our prior belief in $c(S_1,S_5)$ was $P_r(c(S_1,S_5)) = 0.54$, so that $P_r(c(S_1,S_5)|\neg r(S_1,S_5)) \ll P_r(c(S_1,S_5))$. We can also conclude that after the expert inspects a false inference, $P_r(c(S_1,S_6)|\neg r(S_1,S_5)) = 0.95^2\cdot 0.0451 \approx 0.04$ that approaches 0. From Figure 4 and Figure 5, we can observe that the belief in the closely related output relation $c(S_1,S_6)$ drops from 0.51 to 0.041, while the belief in the unrelated relation $c(S_1,S_3)$ remains unchanged at 0.57. As a result, the entire family of false output relations $c(S_1,S_5)$ and $c(S_1,S_6)$ drops in the ranking.

In summary, given an analysis and multiple users' comments to be analyzed, we take as input the set of tuples and grounded clauses produced by the logic relation analysis rule at a fixpoint, and construct the belief network. Next, it performs Bayesian inference to compute the probability of each inference, and presents the output relation with the highest probability for the expert to inspect. The expert then indicates its ground truth, and the algorithm incorporates the expert's feedback as evidence for subsequent iterations. We summarize this process in Figure 6. There are several possible stopping criteria by which the expert could cease interaction. She could choose to only inspect output relations with confidence higher than some threshold p_0, and stops once the confidence of the highest ranked alarm drops below p_0. Alternatively, s/he could choose to only inspect n output relations, and stops after n iterations.

4.2. Learn the Probability P_r for Each Output Relation $c(s_i, s_j)$

In this section, we discuss how to learn the probability P_r for each output relation $c(s_i, s_j)$.

4.2.1. Preliminaries

Before we introduce how to view the derivation graph as Bayesian networks and how to compute P_r for each output relation, we start with some preliminaries.

Rank	Belief	Output relation
1	0.95	$c(S_1, S_1)$
2	0.57	$c(S_1, S_3)$
3	0.54	$c(S_1, S_5)$
4	0.51	$c(S_1, S_6)$

Figure 4: List of the output relation produced with belief before the feedback $\neg r(S_1, S_5)$

Rank	Belief	Output relation
1	0.95	$c(S_1, S_1)$
2	0.57	$c(S_1, S_3)$
3	0.043	$c(S_1, S_5)$
4	0.041	$c(S_1, S_6)$

Figure 5: List of the output relation produced with belief after the feedback $\neg r(S_1, S_5)$

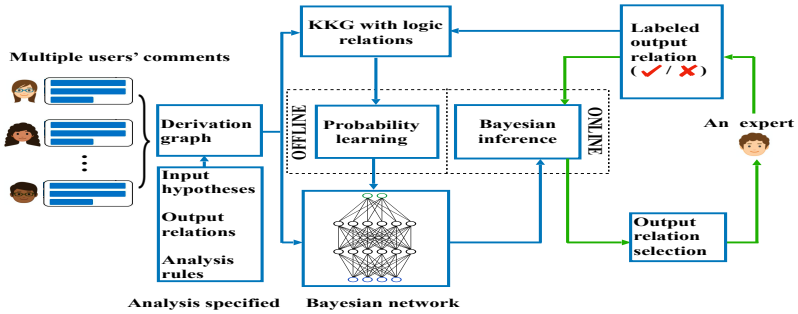

Figure 6: The workflow of the algorithm and the interaction model.

Appending c and GC Assume we have a collection $\{R\}$ of relations. Each relation R has an arity r with a set of tuples $R(v_1, \cdots, v_r)$. The examples of the atoms include the relation c and H from Table 1. The analyst divides these into input hypothesis and output relations, and specifies the computation of the output relations using a set of rules, each of which is of the following form:

$$R_h(u_h) := R_1(u_1), R_2(u_2), \cdots, R_p(u_p), \tag{10}$$

where R_h is an output relation, and $u_h, u_1, u_2, \cdots, u_p$ are free tuples. Examples include the analysis rule r. The free variables of a rule yield a Horn clause, i.e. $R_1(u_1) \wedge R_2(u_2) \wedge \cdots \wedge R_p(u_p) \rightarrow R_h(u_h)$. To solve an input multiple users' comment, we accumulate the grounded clauses until a fixpoint. Given a valuation I of all input relations, we initialize the set of conclusions with $c := I$, and the grounded clauses to the set of input tuples, $GC := \{\textsc{True} \rightarrow t | t \in I\}$. We repeatedly apply each rule to update c and GC until no new conclusions can be reached. That is, whenever $R_p(u_p)$ occurs in c, we update $c = c \cup \{R_h(u_h)\}$, and $GC = GC \cup \{R_1(u_1) \wedge R_2(u_2) \wedge \cdots \wedge R_p(u_p) \rightarrow R_h(u_h)\}$.

Bayesian networks We will only consider boolean-valued random variables, and specialize our definition for this purpose. Assume we have a set of random variables V and a directed acyclic graph $\mathcal{G}(V, E)$ with the random variables V as its vertices. Given $v \in V$, we write $P_a(v)$ for the set of variables with edges leading to v. Formally,

$$P_a(v) = \{u \in V | u \rightarrow v \in E\}. \tag{11}$$

The conditional probability distribution (CPD) of a random variable v is a function which maps concrete valuations $\mathbf{x}_{P_a(v)}$ of $P_a(v)$ to the conditional probability of the event

v = True, and we denote this as $p(v|\mathbf{x}_{P_a(v)})$. Naturally, the complementary event v = False has a conditional probability $p(\neg v|\mathbf{x}_{P_a(v)}) = 1 - p(v|\mathbf{x}_{P_a(v)})$. The Bayesian network, given by $BN = (V, \mathcal{G}, p)$,is essentially a compact representation of the following joint probability distribution:

$$P_r(\mathbf{x}) = \prod_v p(x_v|\mathbf{x}_{P_a(v)}), \tag{12}$$

where the joint assignment $\mathbf{x}$ is a valuation x_v for each $v \in V$, and $\mathbf{x}_{P_a(v)}$ is the valuation restricted to the parents of v.

4.2.2. From Derivation Graph to Bayesian Networks

Next, we are going to explain how to view the derivation graph as a Bayesian network and then compute P_r. The set of output relations c and the grounded clauses GC from the comments at a fixpoint naturally induce a graph $\mathcal{G}(c, GC)$ over the nodes $c \cup GC$, with an edge from a tuple $t \in c$ to a clause $g \in GC$ whenever t is an antecedent of g, and an edge from g to t whenever t is the conclusion of g. We have already seen portions of this graph in Figure 3. Our goal is to view this graph as a Bayesian network with each node $v \in C \cup GC$ as a random variable, and assign the CPDs for each output relation, such as those in Equations (3-6) to each node. However, observing that although the underlying graph of the Bayesian network is required to be acyclic, the derivation graph may have cycles, as shown in Figure 7.

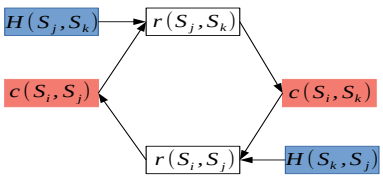

Figure 7: An example about a derivation graph with a cycle

We then choose a subset $GC_c \subseteq GC$ of clauses such that the induced graph is acyclic.

4.2.3. Learning Rule Firing Probability P_r

Given a logic relation analysis rule A and a KG G, the Bayesian network discussed in the previous section is parameterized by the vector of rule probabilities $\mathbf{p}$. To emphasize this dependence, we write $P_{r_\mathbf{p}}$ for the induced joint probability distribution. We will now discuss how we obtain $\mathbf{p}$. With comments of fully labelled data, the rule probability is simply the fraction of times that the rule produces incorrect output relations from true hypotheses. With a less extensively labelled dataset, one faces the challenge of latent (or unobserved) variables, which can be solved with the expectation maximization (EM) algorithm described below.

The maximum likelihood estimator (MLE) For comments and their associated ground truth $\mathbf{v}$, our learning problem is to determine the best weight vector $\mathbf{p}$ which explains $\mathbf{v}$. One common measure of goodness is the likelihood, $\mathcal{L}(\mathbf{p}; \mathbf{v})$, which may be informally motivated as the plausibility of $\mathbf{p}$ given the training data $\mathbf{v}$. Concretely, $\mathcal{L}(\mathbf{p}; \mathbf{v}) = P_{r_\mathbf{p}}(\mathbf{v})$.

To learn the rule firing probabilities is to find the probability vector $\hat{\mathbf{p}}$ with the highest likelihood: $\hat{\mathbf{p}} = \arg\max_{\mathbf{p}} \mathcal{L}(\mathbf{p}; \mathbf{v})$.

MLE by expectation maximization (EM) In our setting, MLE may be performed by a straightforward implementation of the EM algorithm (Koller and Friedman, 2009). Starting with an arbitrary seed $\mathbf{p}(0)$, the algorithm iteratively computes a sequence of estimates $\mathbf{p}(0), \mathbf{p}(1), \mathbf{p}(2), \cdots$, as follows: $p_r^{(t+1)} = \frac{\sum_g P_{r_{\mathbf{p}^{(t)}}}(c_g \wedge A_g)}{\sum_g P_{r_{\mathbf{p}^{(t)}}}(A_g)}$, where A_g is the set of all the antecedents of a grounded clause, and C_g be its consequence. The algorithm guarantees that $\mathcal{L}(\mathbf{p}^{(t)}; \mathbf{v})$ monotonically increases with t, which is also bounded above by 1, so that the procedure converges to a local maximum in practice.

5. The proposed $\mathbf{K^2}$-GNN

In Section 4, we have introduced how to build KKG. Next, we are going to discuss our K^2-GNN framework for multiple comments integration. Since our KKG builds from multiple comments, we can employ a few GNN frameworks for document summarization, which take the KKG as the input. Note that GNN-based document summarization is a well studied problem in natural language processing and our KKG is compatible with a wide range of encoder (with a graph component) and decoder choices in the format of "Encoder → Decoder", e.g, BiLstm+Gnn → Lstm, BiLstm+Gnn → Lstm+Pointer, See et al. (2017) (+ Pointer), and See et al. (2017) (+ Pointer + Coverage). We also take some typical encoder-decoder frameworks, such as BiLstm → Lstm, See et al. (2017) as the baselines for multiple comments integration. Due to the space constraint, the typical decoder frameworks, such as Lstm, Lstm+Pointer, are not discussed here. For the encoder framework, since we combine sequence encoders with GNN, such as BiLstm+Gnn, we are going to discuss the details for this sequential GNN that combines GNN with standard sequence encoders. As input, we take the adjacency matrix A of the KKG and a sequence representation $\mathbf{e} = [\mathbf{e}_1, \cdots, \mathbf{e}_{|V|}]$, where $|V|$ denotes the number of nodes. This sequence representation $\mathbf{e}$ can be modeled by the embedding of a string label of the $|V|$ nodes. For the l-th output, we can construct the sequence GNN by simply computing a simple form of layer-wise propagation $\mathbf{e}_i'^l = \sigma(A^l \mathbf{e}_i^l \mathcal{F}_e^l)$. To obtain a graph level representation, we use the weighted averaging mechanism, where we compute a weight $\sigma(w(\mathbf{e}_i')) \in [0, 1]$ and compute a graph-level representation as $\widehat{\mathbf{e}} = \sum_{i=1}^{|V|} \sigma(w(\mathbf{e}_i'))\mathcal{N}(\mathbf{e}_i')$. We then found that the best results were achieved by computing the final $\mathbf{e}^{l+1}$ as $\mathcal{F}_{e'}^l \mathbf{e}'^l \widehat{\mathbf{e}}$, where $\mathcal{F}_{e'}^l$, $\mathcal{F}_e^l$, $\mathcal{N}$, and w are learnable weights. Besides, $\mathcal{F}_{e'}^l$ and $\mathcal{F}_e^l$ can share the same model weights. When intermediate node annotations $\mathbf{e}$ are not available during training, we treat them as hidden units in the network, and train the whole model jointly by backpropagating through the whole sequence via $-\sum_{\mathbf{e}_i^{l+1} \in \mathbf{e}^{l+1}} \mathrm{softmax}(r(\mathbf{e}_i^{l+1}))$, where $r(\cdot)$ represents the average of ROUGE-1 and ROUGE-2 Recall scores.

6. Experiments

Our experiments mainly answer the following questions: **1)** Are the output inter-sentence logic relations learned correctly by performing the Bayesian inference on the derivation

graph, and what does the proposed KKG look like? **2)** Does the performance improve by leveraging the expert reasoned output relations? **3)** Are the Logic-based inter-sentence relations more significant than other non-logic inter-sentence relations[4]? **4)** Does K^2-GNN scale well to large comments? We will first describe our datasets and experimental setups in subsection 6.1, and then discuss each of the above questions in each following subsection, respectively.

6.1. Datasets and Implementation

We investigate the performance of our K^2-GNN and baselines on a customized CNN/DM dataset (Hermann et al., 2015), using the exact data split provided by See et al. (2017) and a specialized real-world dataset, denoted as SSECIF 200. The CNN/DM data is constructed from CNN and Daily Mail (CNN/DM) news articles along with a few sentences that summarize each article. We group the CNN/DM dataset into C clusters according to their tf-idf cosine similarity between the highlights of any news articles. If the tf-idf cosine similarity in terms of the bag-of-words model is above a threshold of 0.2, we group these news articles into clusters[5]. After preprocessing, we obtain 50 clusters, and each cluster contains nearly 90 documents. We perform each solution on each cluster to integrate the documents in each cluster to generate an integrated document. The SSECIF 200 dataset contains comments on 200 books from Amazon[6]. More Specifically, for each book, comments from 10 participants, with different lengths, have been packaged as a "source document". In both datasets, the ground truth integration of each cluster is generated and verified by professional researchers. For evaluation, we use the ROUGE score metrics with stemming and the stop words are not removed. We use GNNs with the size of a node vector set to $D = 15$ and four hidden layers ($L = 2$). We use a bidirectional LSTM encoder with 1 layer and 256 hidden units, and its sequence GNN extension with 128 hidden units unrolled over 8 timesteps. For the decoder, we use an LSTM with 1 layer and 256 hidden units with attention over the input sequence, and an extension using a pointer network-style copying mechanism. On the CNN/DM dataset, we additionally perform an experiment with the model of See et al. (2017) implemented in OpenNMT (Klein et al., 2017), but using our parameters and proposed attention mechanism. We uniformly assign each rule to a probability of 0.999.

6.2. Learned Probability of Output Relations

The following Figure 8 and Figure 9 show the list of the output relations produced with corresponding beliefs before and after expert's investigation. From Figure 8, we can conclude that the beliefs of the output relations are strictly ranked based on K. For example, the belief of the 1-hop pairs (from the 1st to the 7th in the list) are ranked the highest. The 2-hop pairs (from the 8th to the 11th in the list) are ranked second. The 3-hop pair $c(S_1, S_6)$

4. The examples of non-logic inter-sentence relations include: adjective relations ('adj_sent'), noun relations ('noun_sent'), verb relations ('verb_sent'), pronoun relations ('pronoun_sent'), and co-reference ('coreference'), which present connecting adjectives, nouns, verbs, and pronouns, and co-reference pairs of two sentences, respectively.
5. Note that each news article already contains its highlights in the dataset
6. https://www.amazon.com/

results in the lowest belief. From Figure 9, we can observe that the beliefs of the output relations remain unchanged for the 1-hop nodes, because the expert labels these sets of relations as true. However, when the expert investigates the 8th output relation $c(S_1, S_5)$, the expert labels it as false. Therefore, the belief of the output relation $c(S_1, S_6)$ also drops after $\neg r(S_1, S_5)$ fires. Observe that the expert also labels $c(S_1, S_7)$ and $c(S_1, S_8)$ as false, but labels $c(S_3, S_6)$ as true. That is why $c(S_1, S_7)$ and $c(S_1, S_8)$ drop in the belief rank.

Rank	Belief	Output relation	Rank	Belief	Output relation
1	0.949	$c(S_1, S_2)$	7	0.949	$c(S_5, S_6)$
2	0.949	$c(S_1, S_3)$	8	0.541	$c(S_1, S_5)$
3	0.949	$c(S_1, S_4)$	9	0.541	$c(S_1, S_7)$
4	0.949	$c(S_3, S_5)$	10	0.541	$c(S_1, S_8)$
5	0.949	$c(S_4, S_7)$	11	0.541	$c(S_3, S_6)$
6	0.949	$c(S_4, S_8)$	12	0.513	$c(S_1, S_6)$

Figure 8: List of the output relations produced with belief before the expert's investigation

Rank	Belief	Output relation	Rank	Belief	Output relation
1	0.949	$c(S_1, S_2)$	7	0.949	$c(S_5, S_6)$
2	0.949	$c(S_1, S_3)$	8	0.541	$c(S_3, S_6)$
3	0.949	$c(S_1, S_4)$	9	0.026	$c(S_1, S_5)$
4	0.949	$c(S_3, S_5)$	10	0.015	$c(S_1, S_6)$
5	0.949	$c(S_4, S_7)$	11	0.012	$c(S_1, S_7)$
6	0.949	$c(S_4, S_8)$	12	0.009	$c(S_1, S_8)$

Figure 9: List of the output relations produced with belief after the expert's investigation

6.3. Quantitative Evaluation

We will investigate the performance of our K²-GNN quantitatively, the encoder (with a graph component) and decoder frameworks without KKG, and with KKG are shown in Table 2 and Table 3, respectively. Again, results for models from the literature are obtained after retraining these models with our parameter settings. Across all tasks, the results show the advantage of our KKG and the resulting K²-GNN.

On both datasets, we can see that all K²-GNN augmented models are able to outperform most current advanced methods, such as See et al. (2017) (+ Pointer + Coverage). These K²-GNN augmented models only require a KKG that can be easily obtained in comments tasks via Bayesian inference. On the CNN/DM dataset, K²-GNN gives a better performance than See et al. (2017), since it leverages the expert's logic relation flow. Besides, the experimental results in terms of different encoder and decoder configurations nicely show their effects are mostly orthogonal. Furthermore, the addition of Coverage gives a slightly better performance.

On the CNN/DM dataset, we can observe that See et al. (2017) (+ Pointer + Coverage) together with our proposed K²-GNN achieves the best performance, e.g. 42.4 in ROUGE-1 and 18.2 in ROUGE-2. On the SSECIF 200 dataset, We can observe that our algorithm obtains the highest performance metrics, e.g. 53.0 in ROUGE-1 and 50.0 in ROUGE-2.

To gain a global view of the performance of our K²-GNN, we also compare our approaches with other baseline multi-document summarizers on the CNN/DM dataset. As shown in Table 4, with more fine-grained output relation indicators and our customized human-in-the-loop Bayesian inference, we observe that our K²-GNN significantly outperforms the commonly used baselines and traditional graph approaches, such as Centroid, LexRank, Reg-Sum and RASG. This indicates the advantage of the expert reasoned output relations used in our K²-GNN. Our system also exceeds Reg-Sum (a widely used regression-based summarizer) and RASG (the state-of-the-art multi-document summarizer).

Table 2: Evaluation results for the K^2-GNN on the CNN/DM dataset.

CNN/DM	ROUGE-1	ROUGE-2	ROUGE-L
BiLstm → Lstm w/o K^2-GNN	33.6	11.4	27.9
BiLstm+Gnn → Lstm w/o K^2-GNN	33.0	13.3	28.3
BiLstm+Gnn → Lstm+Pointer w/o K^2-GNN	38.1	16.1	33.2
See et al. (2017) (+ Pointer) w/o K^2-GNN	36.4	15.7	33.4
See et al. (2017) (+ Pointer + Coverage) w/o K^2-GNN	39.5	17.3	36.4
BiLstm → Lstm with K^2-GNN	35.1	11.9	28.9
BiLstm+Gnn → Lstm with K^2-GNN	35.3	14.0	28.9
BiLstm+Gnn → Lstm+Pointer with K^2-GNN	39.7	17.5	35.4
See et al. (2017) (+ Pointer) with K^2-GNN	38.2	17.7	35.8
See et al. (2017) (+ Pointer + Coverage) with K^2-GNN	42.4	18.2	38.2

Table 3: Evaluation results for the K^2-GNN on the SSECIF dataset.

SSECIF 200	ROUGE-1	ROUGE-2	ROUGE-L
BiLstm+Gnn → Lstm+Pointer w/o K^2-GNN	45.0	27.0	37.0
See et al. (2017) (+ Pointer) w/o K^2-GNN	48.0	30.0	41.0
See et al. (2017) (+ Pointer + Coverage) w/o K^2-GNN	49.0	34.0	44.0
BiLstm+Gnn → Lstm+Pointer with K^2-GNN	48.0	37.0	44.0
See et al. (2017) (+ Pointer) with K^2-GNN	50.0	46.0	49.0
See et al. (2017) (+ Pointer + Coverage) with K^2-GNN	53.0	50.0	59.0

Table 4: Comparing our K^2-GNN with conventional multi-document summarizers. The results of our proposed method are in bold.

CNN/DM	ROUGE-1	ROUGE-2
Centroid (Radev et al., 2004)	35.4	16.1
LexRank (Erkan and Radev, 2004)	38.2	16.7
Reg-Sum (Hong and Nenkova, 2014)	38.9	17.0
RASG (Gao et al., 2019)	39.2	17.3
K^2-GNN	**42.4**	**18.2**

7. Conclusions

In this paper, we proposed a new graph framework called K^2-GNN to take advantage of the K-hop knowledge graph (KG) to capture both the intra-sentence and inter-sentence logic relation chains among multiple users' comments. Our extensive experimental results show that our K^2-GNN outperforms all baseline graph models. To gain a global view, K^2-GNN achieves the best performance, e.g., 42.4 in ROUGE-1 and 18.2 in ROUGE-2, comparing with the existing multi-document summarizers. In addition, we show that K^2-GNN uses a reasonable number of interaction runs on both the larger-scale CNN/DM dataset, and the smaller-scale SSECIF dataset.

Acknowledgement Chenyi Hu is partially supported by US National Science Foundation through the grant award OIA 1946391.

References

Danqi Chen, Jason Bolton, and Christopher D Manning. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*, 2016.

Günes Erkan and Dragomir R Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479, 2004.

Patrick Fernandes, Miltiadis Allamanis, and Marc Brockschmidt. Structured neural summarization. *arXiv preprint arXiv:1811.01824*, 2018.

Shen Gao, Xiuying Chen, Piji Li, Zhaochun Ren, Lidong Bing, Dongyan Zhao, and Rui Yan. Abstractive text summarization by incorporating reader comments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6399–6406, 2019.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.

Kai Hong and Ani Nenkova. Improving the estimation of word importance for news multi-document summarization. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 712–721, 2014.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. A survey on knowledge graphs: Representation, acquisition and applications. *arXiv preprint arXiv:2002.00388*, 2020.

Youxuan Jiang, Catherine Finegan-Dollak, Jonathan K Kummerfeld, and Walter Lasecki. Effective crowdsourcing for a new type of summarization task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 628–633, 2018.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*, 2017.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT Press, 2009.

Ni Lao, Tom Mitchell, and William Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 529–539, 2011.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2124–2133, 2016.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. A dependency-based neural network for relation classification. *arXiv preprint arXiv:1507.04646*, 2015.

Diego Marcheggiani and Ivan Titov. Encoding sentences with graph convolutional networks for semantic role labeling. *arXiv preprint arXiv:1703.04826*, 2017.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics*, 5:101–115, 2017.

Dragomir R Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. Centroid-based summarization of multiple documents. *Information Processing & Management*, 40(6): 919–938, 2004.

Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer, 2010.

Huan Rong, Victor S Sheng, Tinghuai Ma, Yang Zhou, and Mznah A Al-Rodhaan. A self-play and sentiment-emphasized comment integration framework based on deep q-learning in a crowdsourcing scenario. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

Sunil Kumar Sahu and Ashish Anand. Drug-drug interaction extraction from biomedical texts using long short-term memory network. *Journal of Biomedical Informatics*, 86: 15–24, 2018.

Sunil Kumar Sahu, Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. Inter-sentence relation extraction with document-level graph convolutional neural network. *arXiv preprint arXiv:1906.04684*, 2019.

Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*, 2017.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. N-ary relation extraction using graph state lstm. *arXiv preprint arXiv:1808.09101*, 2018.

Patrick Verga, Emma Strubell, and Andrew McCallum. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. *arXiv preprint arXiv:1802.10569*, 2018.

Wenhan Xiong, Thien Hoang, and William Yang Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*, 2017.

David M Zajic, Bonnie J Dorr, and Jimmy Lin. Single-document and multi-document summarization techniques for email threads using sentence compression. *Information Processing & Management*, 44(4):1600–1610, 2008.

Jing Zhang, Xindong Wu, and Victor S Sheng. Learning from crowdsourced labeled data: a survey. *Artificial Intelligence Review*, 46(4):543–576, 2016.

Wen Zhang, Bibek Paudel, Liang Wang, Jiaoyan Chen, Hai Zhu, Wei Zhang, Abraham Bernstein, and Huajun Chen. Iteratively learning embeddings and rules for knowledge graph reasoning. In *The World Wide Web Conference*, pages 2366–2377, 2019.