Heuristics for Interpretable Knowledge Graph Contextualization

Kshitij Fadnis[†], Kartik Talamadupula[†], Pavan Kapanipathi[†], Haque Ishfaq[§], Salim Roukos[†], Achille Fokoue[†]

[§] McGill University
School of Computer Science
Montreal, Canada
haque.ishfaq@mail.mcgill.ca

[†]IBM Research
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
{kpfadnis, krtalamad, kapanipa, roukos, achille}@us.ibm.com

Abstract

In this paper, we introduce the problem of *knowledge graph contextualization* – that is, given a specific *context*, the problem of extracting the most relevant sub-graph of a given knowledge graph. The context in the case of this paper is defined to be the textual entailment problem, and more specifically an instance of that problem where the entailment relationship between two sentences P and H has to be predicted automatically. This prediction takes the form of a classification task, and we seek to provide that task with the most relevant external knowledge while eliminating as much noise as possible. We base our methodology on finding the shortest paths in the cost-customized external knowledge graph that connect P and H, and build a series of methods – starting with manually curated search heuristics and culminating in automatically extracted heuristics – to find such paths and build the most relevant sub-graph. We evaluate our approaches by measuring the accuracy of the classification on the textual entailment problem, and show that modulating the external knowledge that is used has an impact on performance.

1 Introduction

Knowledge Graphs (KGs) contain a very large amount of knowledge about the world and phenomena within it. Such knowledge can be very useful in natural language processing (NLP) tasks such as question answering, textual entailment etc. – tasks that can benefit from a large amount of specialized, domain-specific knowledge. However, recent approaches that have tried to use KGs as sources of external knowledge for the textual entailment problem (Wang et al. 2019) have found that bringing in external knowledge from KGs comes with a significant downside – namely noise that is brought in from the external knowledge. This noise mainly occurs due to the fact that KGs are very large graphs that often contain wrong, repeated, and incomplete information. Retrieving a sub-graph of a given KG that is relevant to a given problem instance is a non-trivial task, and continues to be a topic of much research study.

In this paper, we focus on this problem from the perspective of search in the space of graphs. Specifically, we consider the problem of extracting the sub-graph of a given (large) graph that is most relevant to a given context or problem setting – we call this the *knowledge graph contextualization* problem. Obviously, there are many ways of extracting such a sub-graph, and they must all be tied in some way to the overall metric: that is, the performance on the problem setting in question. For the purposes of this study, we fix the problem setting as the textual entailment or natural language inference (NLI) problem, taking after Wang et al. (2019). The textual entailment problem has usually been cast as a classification problem, where a given textual entailment instance consists of a *premise* P and a *hypothesis* H. The label indicates the relationship between H and P.

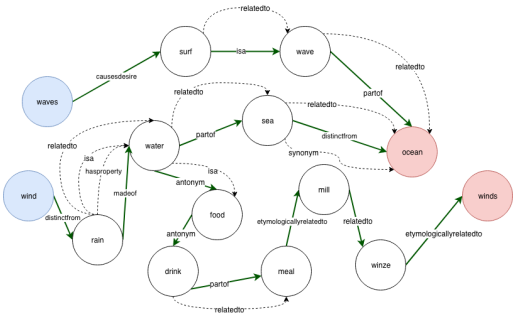

Figure 1: An NLI instance situated in a knowledge graph. Premise nodes are blue, hypothesis nodes red.

The problem with bringing in external knowledge from a knowledge graph is one of scale: for any given entity (node) in the knowledge graph, within a few hops, a large number of nodes are retrieved. Many of these nodes are completely irrelevant to the task at hand, and are not influenced in any way by the context of the problem being solved. Figure 1 shows an example of an NLI problem instance, along with a sub-graph for that instance. The key problem that needs to be solved is one of ranking and filtering the nodes that are retrieved according to some context-sensitive measure. In this paper, we aim to use the entities in the premise P and hypothesis H – as well as the paths that connect them in an external KG – to do this filtering. In brief, our method is as follows: first, we generate the Cartesian product of all pairs of entities $C = P \times H$; then, for each pair in C, we compute the shortest path between the premise entity and the hypothesis entity. The computation of the shortest path is done over a copy of the ConceptNet graph – however, we evaluate various cost

functions to predict the closeness of entities (nodes) in the ConceptNet graph. Each heuristic gives rise to a different, cost-customized copy of the graph, in the following manner: we keep the structure of the graph unchanged, but add a weight to each edge that is computed using a specific cost function. In this way, we invert the traditional notion of the heuristic as used in A* search (Hart, Nilsson, and Raphael 1968) – instead of assigning cost to each node in the graph, we transfer that cost on to teach out-going edge of the node. We evaluate various cost functions that change the nature of the shortest path between two given entities in a KG, and test the knowledge that is thus retrieved for any given pair $\langle \mathrm{P}, \mathrm{H} \rangle$ via performance on the textual entailment problem.

2 Related Work

2.1 Natural Language Inference

Early work on the NLI problem was limited by the availability of small data only, and mostly relied on hand-crafted features (Androutsopoulos and Malakasiotis 2010). To address this problem, Bowman et al. (2015) introduced the large-scale SNLI dataset for NLI, and proposed an LSTM-based neural network model which was the first generic neural model without any hand-crafted features. Bowman et al. use their LSTM model to encode the premise and hypothesis sentences, whose concatenation is then fed to a perceptron classifier. In addition to LSTM-based models, several other neural network models were used for sentence encoding such as GRU (Vendrov et al. 2015), tree-based CNN (Mou et al. 2015), self-attention network (Shen et al. 2018), and BiLSTM (Liu et al. 2016). "Matching aggregation" approaches, on the other hand, exploit various matching methods to obtain an interactive premise and hypothesis space. For example, Wang and Jiang (2015) perform a word-by-word matching of the hypothesis with the premise using match-LSTM (mLSTM). Rocktäschel et al. (2015) use a weighted attention mechanism to get an embedding of the hypothesis conditioned on the premise. Parikh et al. (2016) decompose the entailment problem into sub-problems through an intra-sentence attention mechanism, and are thus able to parallelize the training process. Ghaeini et al. (2018) encode both the premise and the hypothesis conditioned on each other, using BiLSTM and then a soft-attention mechanism over those encodings.

2.2 Knowledge Graphs and NLI

Although there have been extensive studies on the NLI task, the potential for exploiting external knowledge encoded in knowledge graphs (KGs) has not been explored in enough detail. Among the few existing approaches, Chen et al. (2018) use WordNet (Miller 1995a) as the external knowledge source for NLI. They generate features based on WordNet using the relationships in it. However, WordNet, being a lexical database, possesses very few linguistic relationships among entities, and thus its richness as an external knowledge source is limited. There are other KGs such as DBpedia, Yago (Fabian et al. 2007), Freebase (Bollacker et al. 2008) etc. that have become popular due to their expressiveness and the richer information contained in them. One issue with expressive KGs such as DBpedia and ConceptNet (Liu and Singh 2004; Speer, Chin, and Havasi 2017) is that they are quite massive in terms of the nodes and edges contained in them, which makes it hard to extract relevant information useful for the entailment task. However, in spirit, the closest approach to our current work is that of Wang et al. (2019).

3 Methodology

First, we describe the novel methodology that we propose in this paper. The core of our approach is motivated from the understanding that knowledge graphs (KGs) like ConceptNet are essentially directed graphs with labeled edges – the labels denote the relations between the two nodes connected by the edge, while the nodes themselves denote entities. We posit that one of the keys to correctly classifying instances of the textual entailment task is the relationships between the various entities involved in the two propositions. Identifying these relationships using only the text content of the entailment task is an approximate reconstruction of the underlying relationships. While embedding-based methods (see Section 2) situate the sentences in some implicit knowledge-enhanced context, we seek to situate them in a much more explicit graphical context.

In brief, we do this as follows: first, we create different versions of the ConceptNet knowledge graph that feature customized costs as the weights on the relation-edges – we call these *customized cost graphs*. Following this, for each labeled premise and hypothesis pair in the `dev` partition of the `SciTail` dataset, we extract the entities from each respective sentence. We then take the Cartesian product of the premise and hypothesis entities (respectively) to create ordered premise-hypothesis entity pairs. We then find the shortest path between each of these entity pairs in the customized cost graphs. For each premise-hypothesis sentence pair (that is, a textual entailment problem instance), the collection of shortest paths thus found is then associated with the corresponding label for purposes of learning how to predict the entailment accurately (described in more detail in Section 4). In the rest of this section, we provide the details of the process that we have just described.

3.1 Customized Cost Graphs

The first step towards constructing that explicit graphical context is to pick the external knowledge repository. In this paper, we pick the ConceptNet (Speer, Chin, and Havasi 2017) graph, which contains crowdsourced and expert-created knowledge in the form of *entities* (which are represented by nodes in the graph) and *relations* (which are represented by edges in the graph). Typically, the relations (edges) in ConceptNet carry labels which denote the semantic meaning of that edge. These edges are accompanied by a *weight* The central contribution of our work is to redefine these weights along the edges to take into account the structure of the graph. More specifically, we create copies of the ConceptNet knowledge graph and replace the default weights with customized weights on the relation-edges.

3.2 Cost Heuristics

Our quest to retrieve the right knowledge that is useful in classifying a textual entailment instance is grounded in a simple insight: not all relations between entities are equal. Put another way, the ConceptNet graph – which is made up of entities and the relation edges that connect them – needs to be re-weighted in order to reflect this fact. This re-weighting happens by rewriting the weights on the edges of the graph, and treating those weights as a *cost* that is incurred any time that specific edge has to be traversed. In the following, we detail four different heuristics that we use to generate these edge-costs: We call each of the copies of ConceptNet thus produced as a *cost graph*, and demonstrate the use of these various cost graphs in Section 3.4.

Default Cost (DC) This is the simplest case we consider, where we assign every single edge in our target graph (ConceptNet) a cost of 1.0. This essentially turns the path-finding problem between two given nodes on the graph into a problem of minimizing the number of hops: the shortest hops give the most efficient path.

Relevant Relations (RR) The next obvious step in defining costs is to consider the case where some relations are different from others: that is, some relations are more important to the task at hand than others. Specifically, in the case of this work, we look at relations that we consider *relevant* to the textual entailment task. This is a manually filtered subset of the total list of relations present in ConceptNet. Some examples of relations that are included in this subset are RELATEDTO, IS-A, SIMILARTO, DERIVEDFROM etc. For each of these relations, the edge costs of any instance of that relation in the graph is reduced, thus reducing the cost of taking such an edge, and encouraging a shortest-path search algorithm to consider these edges first.

Relation Frequency (RF) The two prior heuristics feature values that are manually decided and set: that is, we determine on our own what the weight on an edge should be. The next step up in complexity is to automate the computation of that weight, and base that computation on some feature of the graph itself. The first such heuristic is to simply count the frequency of the relations as they occur in relation to an entity. We specifically implement this heuristic as the normalized count of the number of outgoing edges bearing the same relation name from a given node. That is, given a node n that represents an entity in the graph, and $rel(n)$ the set of outgoing edges from n, we represent the cost c_i for an edge $e_i \in rel(n)$ as $c_i = \frac{|e_i|}{|rel(n)|}$. For example, consider a node n_1 that has three outgoing edges: $\{e_1, e_2, e_1\}$. Using the above formula, the weights of the e_1 edges would be set to $c_{e_1} = 0.67$, while the edge e_2 would have a cost of 0.33. This ensures that the edge that is "rarer" is given a lower cost, and is favored by a shortest-path algorithm in case there is more than one way to travel from node n_1 to a neighboring node.

Global Relation Frequency (GRF) The final heuristic that we consider builds on top of the relation frequency metric by addressing a significant issue: the presence of common relations that occur throughout the knowledge graph, but may occur relatively fewer times at any one individual node. An example of such a relation is IS-A; while this relation is likely to occur relatively fewer times at any given node, it is clear that it occurs throughout the graph. We want to ensure that a truly rare relation that participates in an entailment instance is thus given more importance (and subsequently less cost) than one which occurs throughout the graph. To do this, we follow the inspiration of TF-IDF (Salton and Buckley 1988), which is often used to address similar issues in text corpora.

We first compute the Inverse Node Frequency (INF) (the analog of IDF) for every relation in the graph. Given a graph with node-set N, let the quantity n_{rel_i} be the number of times relation rel_i appears in the nodes in N as an outgoing edge. The INF for edges with the relation label rel_i can then be calculated as $INF_{rel_i} = \log \frac{|N|}{n_{rel_i}}$. Next, we compute the normalized Relation Frequency (RF) as in the previous section. Thus given a node $n \in N$ with a set of outgoing edges e, the RF for an edge with relation i can be calculated as $RF_{rel_i} = \frac{|e_i|}{|e|}$. Since we are interested in promoting "rarer" relations by associating lower cost with them, we invert INF during the the calculation of the final cost metric, giving us the cost as $c_i = RF_i \times \frac{1}{INF_i}$.

3.3 Ordered Premise & Hypothesis Pairs

Once we generate the various cost graphs as described above, it is then time to use those respective graphs to obtain the relationships between the two sentences in a given textual entailment instance. As before, let us assume that this instance is denoted $\tau = \langle p, h \rangle$, where p is the premise sentence and h is the hypothesis sentence. The first step we take is to represent each sentence using its respective entities: that is, we collapse the representation of a sentence into an ordered set of those entities from the sentence that also appear in ConceptNet.[1] Let us denote these ordered sets as P and H respectively. Since we do not know which entities in the premise and which ones in the hypothesis contribute directly to the classification of the entailment relationship, we take the cartesian product of the two ordered sets P and H to generate the set of all possible ordered pairs between p and h. This set $S = \text{P} \times \text{H} = \{(a,b) \mid a \in \text{P},\ b \in \text{H}\}$ is then used as the input for the shortest path generation step.

3.4 Shortest Paths

Once we have the sets of premise-hypothesis entity pairs from Section 3.3, we move on to finding all shortest paths between the first and second entity of each pair, for every cost graph outlined previously. We employ `NetworkX`'s (Hagberg, Swart, and S Chult 2008) implementation of the Dijkstra shortest-path algorithm. Since ConceptNet has about 1 million nodes and well over 3 million edges, finding shortest paths is an extremely expensive process. Additionally, after an analysis of entity pairs from Concept-

[1] We perform stemming in order to turn words into their normative forms, before doing a look-up in the list of ConceptNet entities.

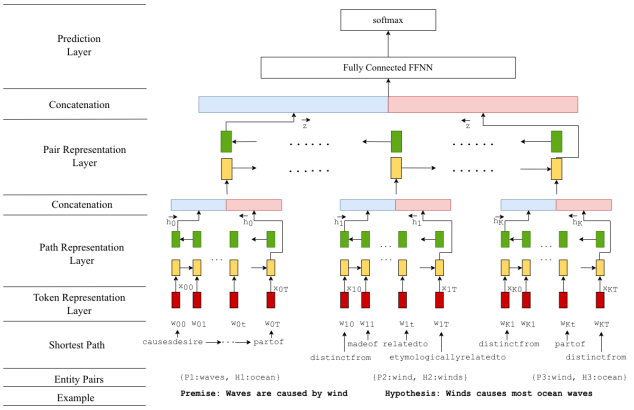

Figure 2: Architecture of our GRN network.

Net that feature more than one direct edge between them (multi-edges), we find that the most common relationship (`RelatedTo`) occurs about 83% of the time. The second most common relationship (`FormOf`) occurs in about 33% of cases. Further, these two relations co-occur around 30% of the time, and of those cases, for about 97% of the time, they are the *only* two relations connecting that entity pair. All of these support our hypothesis that selecting at random between paths that contains either of these relationships will not have a significant impact on the NLI classification problem. We therefore reduce the problem of finding all shortest paths between premise-hypothesis entity pairs to one of finding *a single* shortest path.

4 Using KG Information via Shortest Paths

Once the pairwise shortest paths are generated, we need to use them in a way that enables us to train on labeled textual entailment instances, in order to make predictions on new instances. In this we focus particularly on the *path* part of the shortest paths – that is, we are interested in considering the relations used to connect a given premise and hypothesis pair from a textual entailment instance. This harks back to our hypothesis in Section 3 that the relationships between entities in the textual entailment instance are key to identifying the overall entailment relationship. In this section, we detail two specific ways in which we use the shortest paths: by accounting for the number of times relations appear in those paths; and then the sequence order in which they appear. These two approaches are in contrast to the work of Wang et al. (2019), as they only consider entity-level information and completely ignore relationships.

4.1 Text Model: mLSTM

Most models for the NLI problem use only the premise and hypothesis sentence as input; due to this fact, we decided to use match-LSTM (mLSTM) (Wang and Jiang 2016) as our text-based model. The specific implementation of mLSTM that we use encodes both premise and hypothesis as Bi-GRUs (as against Bi-LSTMs), and a fixed representation of the hypothesis that is premise-attended is output. Such asymmetry in the modeling of the premise-hypothesis relationship has led to an improved performance of mLSTM on various leaderboards.

4.2 Relationship Frequency Vectors

In order to enhance the text models that have been used by prior work, we incorporate external knowledge in the form of the frequency distribution of relations present along the shortest paths between premise-hypothesis entity pairs. The size of the vector representing the paths is same as the number of distinct relationships in the knowledge graph. In our case, since we use ConceptNet, it has 47 distinct relationships. Hence, each relationship is assigned a fixed positional index in this vector.

We calculate the frequencies of relations present in the paths across all premise-hypothesis entity pairs in a single NLI instance. For example, consider that we have two premise-hypothesis entity pairs with shortest paths RELATEDTO $\rightarrow$ ISA $\rightarrow$ RELATEDTO; and RELATEDTO $\rightarrow$ SYNONYM $\rightarrow$ FORMOF respectively. The frequency counts would then be RELATEDTO: 3, ISA: 1, SYNONYM: 1, FORMOF: 1; and 0 everywhere else. The non-zero frequency values are set at their respective relation position index. The relation frequency vector thus formed is concatenated with the final hidden state from the text model, and the combination is then forwarded to a fully connected feed forward network.

We experimented with scaling the relation frequency vector to higher dimensions via linear layers; these results are reported in Table 3. The use of this simple frequency-based model makes it possible for us to analyze the learned weights, and subsequently intuit the importance and contribution of each relation in the classification task accuracy.

4.3 Recurrent Neural Networks

After modeling the shortest paths as the frequency counts of the relations along those paths, the next obvious step is to use the sequentiality inherent in a shortest path as well. Recent work on Graph Convolutional Recurrent Networks (GCRN) (Seo et al. 2018) has explored representing sequential graphical structures as fixed representations. One of the major difference between that approach and the one we take in this work is the degree or *level* of sequentiality. In our current problem, we are faced with two levels of sequential information. One of these is at the level of ordered premise-hypothesis entity pairs. The other is at the level of the path, which is represented as a sequence of relations, entities, or both; per premise-hypothesis entity pair.

We first describe how we process the shortest paths to capture the bi-level sequentiality inherent in them. As before, we assume each textual-entailment instance τ consists of premise (p) and hypothesis (h), which together constitute a sentence pair. After processing each τ as outlined in Sections 3.3 and 3.4, we obtain an ordered set of shortest paths. Each of these shortest paths can be represented by either the entities along that path (alone), the relations along that path (alone), or a combination of the entities and relations both. Our work follows various hierarchical architectures that have been proposed for different learning-centric tasks (Sordoni et al. 2015; Li, Luong, and Jurafsky 2015; Yang et al. 2016; Serban et al. 2016). The hierarchical assumption formulates a sequence at two levels: (1) a sequence

of tokens for each pair; and (2) a sequence of pairs. We model this hierarchy as two recurrent neural networks.

Figure 2 shows the architecture of our *Graph Recurrent Network* (GRN) architecture. We describe the functioning of the GRN via a simplified working example. Consider the two sentences: WAVES ARE CAUSED BY WIND (premise); and WINDS CAUSES MOST OCEAN WAVES (hypothesis). As described in Section 3.3, we first find all possible premise-hypothesis entity pairs. This particular example gives us 12 such pairs: 3 premise (WAVES, CAUSED, WIND) times 4 hypothesis (WINDS, CAUSES, OCEAN, WAVES) entities. We further simplify for the sake of exposition and focus on three entity pairs: (WAVES, OCEAN), (WIND, WINDS), and (WIND, OCEAN). As explained in Section 3.4, we identify shortest paths for each of these pairs. For example, for the pair (WAVES, OCEAN), the shortest path looks like: WAVES $\rightarrow$ CAUSESDESIRE $\rightarrow$ SURF $\rightarrow$ ISA $\rightarrow$ WAVE $\rightarrow$ PARTOF $\rightarrow$ OCEAN, where WAVES, SURF, WAVES and OCEAN are entities along the path; and CAUSESDESIRE, ISA and PARTOF are the relationships connecting them in sequential order.

The GRN model can take either relations, entities, or relations plus entities as its input. In Figure 2, we show an instance where relations are fed as input to the token-representation layer. At this point, the tokens – which are relations in this case – are transformed into vector representation using an embedding matrix. The transformed representations are then fed to a bidirectional Recurrent Neural network (RNN) in the sequence order captured by the shortest path. The final hidden states from the bidirectional RNN are then concatenated to form a representation for the whole path. Thus after passing through the path representation layers, we have vector representations for each of the entity pairs. These representations are then fed into a second bidirectional RNN in the order prescribed by the ordered set of entity pairs. Once the final hidden states of the pair-level encoder are concatenated, a feed-forward network with rectified linear units (ReLU) and linear activation with softmax layer is used as a final prediction layer.

Token-level Encodings Each pair $pair_i$ consists of a sequence of tokens $w_{it}, t \in [0,T]$ which are embedded using an embedding matrix W_t as $x_{it} = W_t w_{it}$. Then the bidirectional token-level RNN – a GRU (Cho et al. 2014) in our case – is used to form a fixed length representation by concatenating the final state from forward($\overrightarrow{h_{it}} = \overrightarrow{GRU}(x_{it}), t \in [1,T]$) and backward ($\overleftarrow{h_{it}} = \overleftarrow{GRU}(x_{it}), t \in [T,1]$) passes in the GRU. This yields $h_i = [\overrightarrow{h_{iT}}, \overleftarrow{h_{i0}}]$. Note that we use ComplEx (Trouillon et al. 2016) and TransH (Han et al. 2018) knowledge graph embeddings for token-level embeddings. These emebeddings are trained on ConceptNet using OpenKE [2].

Pair-level Encodings: The input to the pair-level encoder is a sequence of token-level representations $h_1, h_2, \ldots, h_K$. Then, just as above, a bidirectional GRU computes the fixed length representation as: $\overrightarrow{z} = \overrightarrow{GRU}(h_k), t \in [1,K]$; $\overleftarrow{z} = \overleftarrow{GRU}(h_k), k \in [K,1]$; and $z = [\overrightarrow{z}, \overleftarrow{z}]$.

[2] https://github.com/thunlp/OpenKE

5 Experimental Setup

In this section, we talk about our experimental setup, which includes the dataset and knowledge graph used; the implementation of that knowledge graph; the computational power used for our experiments; and various initializations and hyperparamters. We list all of these to bolster the reproducibility of our work.

5.1 Dataset & Knowledge Graph

In order to evaluate our approach, we use SciTail (Khot, Sabharwal, and Clark 2018), which is a science domain entailment dataset. The SciTail dataset was created from a corpus of science domain multiple choice questions for $4th$ and $8th$ grade. It has approximately $28K$ premise-hypothesis pairs, which are divided into train, dev, and test sets. The main motivation behind using SciTail is the ability to use it for additional downstream NLP tasks like question answering.

There are multiple open knowledge sources available such as DBpedia (Auer et al. 2007), WordNet (Miller 1995b), and ConceptNet (Speer, Chin, and Havasi 2017). Each knowledge source contain different kinds of information. For example, DBpedia is fact based and comprises information relating Wikipedia entities; WordNet is a linguistic knowledge base; and ConceptNet contains common sense information gathered by crowdsourcing. Thus selecting the right knowledge source for a task or dataset is non-trivial. Wang et al. (2019)'s work provides some guidance on this task, by evaluating the relevance of each of these knowledge bases to the SciTail dataset; their conclusion is that ConceptNet is the best KG for the SciTail dataset.

5.2 Graph Implementation

ConceptNet 5.6 (Speer, Chin, and Havasi 2017) consists of a total of $32,755,210$ entries, capturing concepts and relations spanning over more than 83 languages. For this work, we only focused on the $3,098,578$ English language entries. We transformed these entries into commonly used graph format of $\langle$ subject, predicate, object $\rangle$ tuples. This reformatted ConceptNet was represented as a `MultiDiGraph` – due to the existence of multiple relations (edges) between some entities – with the `NetworkX` (Hagberg, Swart, and S Chult 2008) Python library.

5.3 Compute Power

The ConceptNet filtering, cost graph customization, and shortest path generations were performed on a 56 core Intel(R) Xeon(R) CPU E5-2683 v3 @ 2.00GHz machine. The RNN and GRN models were trained and evaluated using Tesla P100 Nvidia GPUs with 16GB of memory.

5.4 Initializations & Hyperparameters

In the Graph Recurrent Network (GRN) model, we used ComplEx (Trouillon et al. 2016) and TransH (Han et al. 2018) knowledge graph embeddings for the token-level encoder, with the embedding dimension set to 300. The token-level and pair-level encoders used single-layered bidirectional GRUs with a hidden size of 300. Parameters were not

	GRN				GRN + mLSTM				mLSTM
	DC	RR	RF	GRF	DC	RR	RF	GRF	
Relations Only	59.27	59.43	60.10	60.90	87.88	87.70	88.26	88.42	88.42
Entities Only	67.65	63.57	63.88	64.95	87.19	86.73	86.80	87.42	88.42
Relations + Entities	64.11	65.72	64.18	64.26	87.26	87.65	86.42	**88.65**	88.42

Table 1: Accuracy values for GRN experiments; embeddings are GloVe, ComplEx, 300D.

	GRN				GRN + mLSTM				mLSTM
	DC	RR	RF	GRF	DC	RR	RF	GRF	
Relations Only	60.65	57.44	59.58	63.34	87.58	88.26	87.42	87.34	88.42
Entities Only	65.87	63.65	65.03	63.80	86.04	86.58	85.19	86.27	88.42
Relations + Entities	63.04	65.57	64.57	59.97	86.57	86.66	86.50	87.65	88.42

Table 2: Accuracy values for GRN experiments; embeddings are GloVe, TransH, 300D.

shared between token-level and pair-level encoders. A two-layered fully-connected feed-forward neural network with ReLU and linear activation, and dropout of 0.2 and 0.0 respectively was used for the prediction layer. The size of the hidden layer was set to 200.

Our models were implemented with `AllenNLP`, a popular NLP library. We tuned the hyperparameters for the models using the validation set. We used a sigmoid function and minimized cross-entropy loss for training and updating the model. The training cycle involved a 150 epoch run, with a 20 epoch patience cutoff. The batch size was set to 64, and gradients were clipped at 5.0. The trainer was configured to use the Adam (Kingma and Ba 2014) optimizer with a learning rate of 0.001.

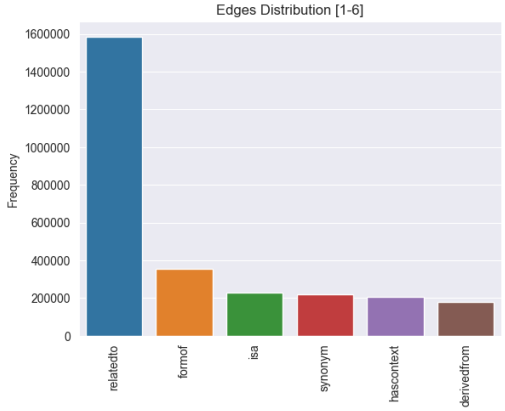

Figure 3: Distribution of edges in ConceptNet, top 6 edges.

6 Results

In this section, we outline our results. The section is split into three parts: we first look at various graph-related statistics across the various customized cost graphs to externalize what the cost customization does to the retrieval of context-relevant knowledge. We then look at the performance of our classification methods on the NLI problem, both quantitatively and qualitatively.

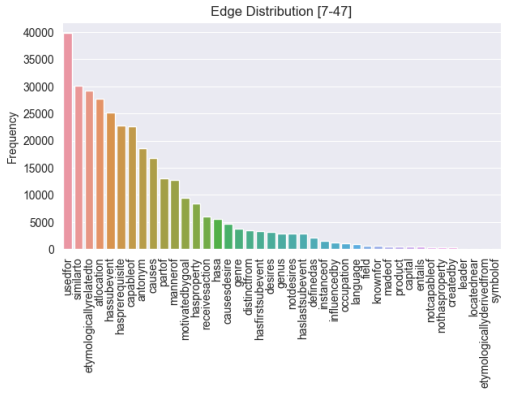

Figure 4: Distribution of edges in ConceptNet outside the top 6.

	DC	RR	RF	GRF
Frequency Only	87.73	87.57	87.88	87.88
Transformation	89.03	87.35	87.73	87.27

Table 3: mLSTM + Frequency Vector Accuracy.

6.1 Graph Statistics

We start by looking at the knowledge extracted by different heuristic cost functions for the same premise-hypothesis samples. A multi-edge directed graph is constructed by combining entity and relation information along the shortest paths for all entity pairs in every premise-hypothesis sample. The numbers shown in Table 4 are averaged over graphs for all premise-hypothesis samples. As we employ more informed cost heuristic functions, there is a steady increase in the number of nodes and edges per multi-edge directed graph. The higher in-degree and out-degree for these graphs indicates the increased variation in the set of nodes that can be reached from a given node, and the set of nodes from which a node can be reached respectively. This trend validates the informativeness in terms of diversity for the RF and GRF cost functions as compared to RR and DC.

The reduction in premise-hypothesis samples that feature pairs with more that one shortest path between them might

	DC	RR	RF	GRF
Nodes	70	72	87	92
Edges	190	196	380	415
In-degree	2.78	2.77	4.24	4.34
Out-degree	2.7	2.69	4.12	4.34
>1 path	23501	23504	234	39

Table 4: Graph statistics for the various heuristic cost functions.

	1st Position	2nd Position	Last Position
Rank 1	RelatedTo	RelatedTo	RelatedTo
Rank 2	FormOf	FormOf	Synonym
Rank 3	Synonym	Synonym	IsA
Rank 4	IsA	IsA	SimilarTo
Rank 5	SimilarTo	HasContext	UsedFor

Table 5: Relation frequency ranks for paths of length 2 to 8.

seem out of the ordinary at first glance, but there is a very good reason for such a reduction. This happens due to the discrete versus continuous nature of the cost functions. We know from Section 3.2 that DC and RR assign a fixed value cost by counting edges, whereas RF and GRF compute floating values for cost. This results in a very low likelihood that the latter cost functions will lead to the exact same (summed) cost across two different paths.

6.2 Quantitative Results

Table 1 shows the performance of the GRN and mLSTM + GRN models across different heuristic cost functions (DC, RR, RF, GRF) and external information types (relations, entities, relations + entities). We observe that the GRN model by itself cannot match the performance of a text model (mLSTM); simultaneously, the GRN + Text model only managed to marginally improve performance accuracy with the GRF heuristic and external knowledge consisting of both relations & entities. In the case of GRN models, even though the overall accuracy is not comparable with the mLSTM text model, we see a consistent gain in performance in most of the cases where an informed heuristic like GRF is used instead of RR or DC. This trend is also observed in GRN + Text models. This indicates that graph structure based heuristics like RF and GRF are better at capturing more relevant external knowledge. We also explored TransH (Han et al. 2018) graph embeddings, but these models did not perform very well; this is reflected in Table 2. However, we present these results in the spirit of full disclosure, in the hope that they will assist future research.

With respect to the knowledge graph itself, we noticed some peculiar characteristics in ConceptNet. The number of relationships in ConceptNet (47) are extremely small when compared to the number of unique entities ($\approx$ 1 million). This causes certain sets of relations to be repeated quite frequently, thus losing all uniqueness. As Figure 3 shows, the top 6 edges – RELATEDTO, FORMOF, ISA, SYNONYM, HASCONTEXT, DERIVEDFROM – are repeated more than $150,000$ times; with the most frequent relation occurring more than 1.5 million times. In contrast, the remaining 41 relations appear relatively infrequently, as shown in Figure 4. This skewed distribution leads to the Top-6 most frequent relations dominating the shortest paths for the majority of premise-hypothesis entity pairs. An additional piece of evidence in support of this argument is presented in Table 5, which shows the most popular (top 5 ranks) relations for the first, second, and last[3] positions across all paths of length 2 to 8. These positions are mostly dominated by relations from Figure 3. This frequency-based relation domination thus overwhelms any real signal that might be coming from other paths that contain more infrequent and unpopular relations. This is an extremely interesting result for future work.

6.3 Qualitative Results

In this section, we highlight the promise of our approach looking at examples that are classified correctly by either the mLSTM approach (here called text), or the mLSTM + GRN approach (here called graph). We compared predictions from the text model against the graph model. Overall, we noticed that the graph model was able to handle sentences with higher numbers of entity pairs, thus resulting in graphs with a higher number of nodes and edges. Out of 46 instances that the graph gets right (but not text), 7 contained over 800 entities; while out of the 49 instances that text got right (but not graph), only 2 had those many entities. If we drop the threshold to 300 entities, these numbers flip to 19 for text and 13 for graph. This aligns well with the common problem faced by text-only models, which fail over long sequences of texts. This also shows the value of our graph approach, which is able to explicitly incorporate external knowledge and scale for instances that are more complex.

7 Conclusion

In this paper, we presented the notion of *contextualizing* a knowledge graph by customizing the edge-weights in that graph with costs produced by various heuristic functions. We used these *cost-customized* graphs to find different shortest paths for different instances of the NLI problem, and trained two different classifiers using the sequence information from the shortest paths. Our results show some clear avenues for immediate future work, including: (1) testing on other KGs and NLI datasets; (2) experimenting with other, more complex cost functions; and (3) automating the construction of the cost function via reinforcement learning.

References

[Androutsopoulos and Malakasiotis 2010] Androutsopoulos, I., and Malakasiotis, P. 2010. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research* 38:135–187.

[Auer et al. 2007] Auer, S.; Bizer, C.; Kobilarov, G.; Lehmann, J.; Cyganiak, R.; and Ives, Z. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*. 722–735.

[3] We choose the first and last because of the bidirectional nature of the encoding approach that we pursue.

[Bollacker et al. 2008] Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 1247–1250. AcM.

[Bowman et al. 2015] Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

[Chen et al. 2018] Chen, Q.; Zhu, X.; Ling, Z.-H.; Inkpen, D.; and Wei, S. 2018. Neural natural language inference models enhanced with external knowledge. In *Proceedings of ACL 2018*.

[Cho et al. 2014] Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[Fabian et al. 2007] Fabian, M.; Gjergji, K.; Gerhard, W.; et al. 2007. Yago: A core of semantic knowledge unifying wordnet and wikipedia. In *16th International World Wide Web Conference, WWW*, 697–706.

[Ghaeini et al. 2018] Ghaeini, R.; Hasan, S. A.; Datla, V.; Liu, J.; Lee, K.; Qadir, A.; Ling, Y.; Prakash, A.; Fern, X. Z.; and Farri, O. 2018. Dr-bilstm: Dependent reading bidirectional lstm for natural language inference. *arXiv preprint arXiv:1802.05577*.

[Hagberg, Swart, and S Chult 2008] Hagberg, A.; Swart, P.; and S Chult, D. 2008. Exploring network structure, dynamics, and function using NetworkX. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

[Han et al. 2018] Han, X.; Cao, S.; Xin, L.; Lin, Y.; Liu, Z.; Sun, M.; and Li, J. 2018. Openke: An open toolkit for knowledge embedding. In *Proceedings of EMNLP*.

[Hart, Nilsson, and Raphael 1968] Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* 4(2):100–107.

[Khot, Sabharwal, and Clark 2018] Khot, T.; Sabharwal, A.; and Clark, P. 2018. Scitail: A textual entailment dataset from science question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[Kingma and Ba 2014] Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Li, Luong, and Jurafsky 2015] Li, J.; Luong, M.-T.; and Jurafsky, D. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

[Liu and Singh 2004] Liu, H., and Singh, P. 2004. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal* 22(4):211–226.

[Liu et al. 2016] Liu, Y.; Sun, C.; Lin, L.; and Wang, X. 2016. Learning natural language inference using bidirectional lstm model and inner-attention. *arXiv preprint arXiv:1605.09090*.

[Miller 1995a] Miller, G. A. 1995a. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

[Miller 1995b] Miller, G. A. 1995b. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

[Mou et al. 2015] Mou, L.; Men, R.; Li, G.; Xu, Y.; Zhang, L.; Yan, R.; and Jin, Z. 2015. Natural language inference by tree-based convolution and heuristic matching. *arXiv preprint arXiv:1512.08422*.

[Parikh et al. 2016] Parikh, A. P.; Täckström, O.; Das, D.; and Uszkoreit, J. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

[Rocktäschel et al. 2015] Rocktäschel, T.; Grefenstette, E.; Hermann, K. M.; Kočiskỳ, T.; and Blunsom, P. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

[Salton and Buckley 1988] Salton, G., and Buckley, C. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24(5):513–523.

[Seo et al. 2018] Seo, Y.; Defferrard, M.; Vandergheynst, P.; and Bresson, X. 2018. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*, 362–373. Springer.

[Serban et al. 2016] Serban, I. V.; Sordoni, A.; Bengio, Y.; Courville, A.; and Pineau, J. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[Shen et al. 2018] Shen, T.; Zhou, T.; Long, G.; Jiang, J.; Wang, S.; and Zhang, C. 2018. Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling. *arXiv preprint arXiv:1801.10296*.

[Sordoni et al. 2015] Sordoni, A.; Bengio, Y.; Vahabi, H.; Lioma, C.; Grue Simonsen, J.; and Nie, J.-Y. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 553–562. ACM.

[Speer, Chin, and Havasi 2017] Speer, R.; Chin, J.; and Havasi, C. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*, 4444–4451.

[Trouillon et al. 2016] Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; and Bouchard, G. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, 2071–2080.

[Vendrov et al. 2015] Vendrov, I.; Kiros, R.; Fidler, S.; and Urtasun, R. 2015. Order-embeddings of images and language. *arXiv preprint arXiv:1511.06361*.

[Wang and Jiang 2015] Wang, S., and Jiang, J. 2015. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*.

[Wang and Jiang 2016] Wang, S., and Jiang, J. 2016. Learn-

ing natural language inference with LSTM. In *Proc. of NAACL-HLT*, 1442–1451.

[Wang et al. 2019] Wang, X.; Kapanipathi, P.; Musa, R.; Yu, M.; Talamadupula, K.; Abdelaziz, I.; Chang, M.; Fokoue, A.; Makni, B.; Mattei, N.; et al. 2019. Improving natural language inference using external knowledge in the science questions domain. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7208–7215.

[Yang et al. 2016] Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; and Hovy, E. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 1480–1489.