Reasoning over Procedural Documents

A Project Report

Submitted For

the Degree of

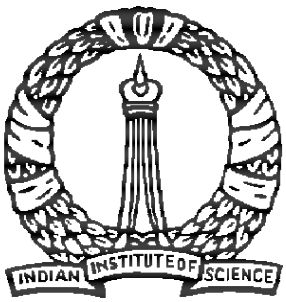
Master of Technology

in

Computational and Data Sciences

by

Vipul Kumar Rathore



Department of Computational and Data Sciences

Indian Institute of Science

BANGALORE – 560 012

June 2019

Declaration of Originality

I, Vipul Kumar Rathore, with SR No. 06-02-01-10-51-17-1-14754 hereby declare that the content presented in the thesis titled: **"Reasoning over Procedural Documents"** represents my original work that I carried out in the Department of Computational and Data Sciences at Indian Institute of Science as a part of Ericsson Research Fellowship during the years 2017-2019. With my signature, I declare that:

- I have not committed any plagiarism of intellectual property. I have clearly mentioned and referenced others work, wherever required.
- I have clearly acknowledged others contributions to my work.
- I have not manipulated any data or results.
- I am aware that any false claim made by me will lead to severe disciplinary action.
- I am aware that the work may be screened electronically for plagiarism check.

June, 2019 Student Signature

As supervisor of the work mentioned above, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Advisor Name: Dr. Partha Pratim Talukdar Advisor Signature

Acknowledgements

I wish to express my profound gratitude and sincere thanks to my advisor, Dr. Partha Pratim Talukdar, for his valuable guidance, patience and immense support throughout the last one year. I am grateful to the almighty for having him as my advisor. It was his energetic personality, dedication for research and strong research ethics that inspired me every moment. I thank the members of MALL Lab for their support in my project, helpful advice and making this journey of bi-directional knowledge and ideas transfer a memorable one. I am grateful to my batch-mates as well for their unconditional support and making learning a richer experience for me. I would like to thank Abhishek Kumar from IIT Patna and Dr. Badrinath Ramamurthy from Ericsson Research for their contributions to my project as well.
I am grateful to the Department of Computational and Data Sciences and MALL Lab for providing me with such high-quality research facility. I would like to thank Ericsson Research, for supporting me financially through their fellowship program throughout my course of study at IISc. Last but not least, I would like to express my heartfelt gratitude towards my parents for their continued support. My masters have been a humbling experience and I realize that there is so much more to learn ahead. I will cherish this phase of my life forever.

Abstract

Procedural Extraction is an important and yet an underexplored problem today in a world having profusion of text that describes complex, dynamic worlds in which entities' relationships evolve through time. This includes news articles, scientific manuals, and procedural text (e.g., recipes, how-to guides, and so on). Reasoning on this kind of text data is implicit but can be useful for automating certain tasks such as automatically filling user form (given minimal user details) from a set of instructions for a procedure like how to apply for Visa for a country or how to register for voter id etc.

Our downstream task of interest is question-answering on documents sampled from biological sources of text, describing natural processes such as photosynthesis, hydroelectricity production etc. This requires tracking the evolving state of entities in the process, wherein the state (property) of interest is the physical location of the entity. For e.g., in photosynthesis process, "light, water and CO_2 combine in the leaf. As a result, the mixture is produced called sugar." In this example, the model should be able to infer that the entities 'light', 'water' and 'CO_2' get destroyed in the physical location namely 'leaf' and a new entity 'sugar' is produced in the 'leaf'.

Note that different tasks can have different properties of interest such as for recipe data, it could be shape, composition, location of ingredients. For our research purpose, we only care about the physical location of an entity as the state of interest. We could do domain adaptation/transfer learning from this property of interest to some other for some other task, as a future line of work.

Contents

List of Tables

List of Figures

Chapter 1

Question Answering on Procedural Documents - Literature Survey

1.1 Introduction

Developing a machine reading comprehension model that is able to read a text document and answer questions has been a crucial problem in NLP and AI domains. However, most of the research in this domain revolves around answering factoid style questions (E.g., Where was the 44th president of United States born?) on top of factual text data (E.g., wikipedia page of Mr. Barack Obama) (Seo et al., 2017a [1]; Clark and Gardner, 2017 [2]), enabled by well-designed datasets and modern deep learning models. However, these models find it difficult to answer implicit questions that require inference (Jia and Liang, 2017 [3]).

Consider the text in Figure 1.1 describing the photosynthesis process.

Chloroplasts in the leaf of the plant trap light from the sun. The roots absorb water and minerals from the soil. This combination of water and minerals flows from the stem into the leaf. Carbon dioxide enters the **leaf**. Light, water and minerals, and the carbon dioxide all combine into a mixture. This mixture forms **sugar** (glucose) which is what the plant eats.

Q: Where is sugar produced?
A: in the leaf

Figure 1.1: *A piece of text from ProPara dataset about photosynthesis (bold highlights the question and answer elements). Procedural datasets are challenging even for human beings because questions (e.g., the one shown here) often require reasoning and inference about the process states.*

Although state-of-the-art models on SQuAD (Rajpurkar et al., 2016 [4]) can answer lookup questions such as: Q1: What do the roots absorb? (A: water, minerals) with high accuracy, they struggle when answers are implicit and demand inference and reasoning, e.g., Q2: Where is sugar produced? (A: in the leaf) In order to answer this, the system apparently needs knowledge of the world and the ability to reason about state transitions across multiple sentences: If CO_2 enters the leaf (explicitly stated), then it will remain at the leaf (not stated), and since it is then utilised to produce sugar, this implies the sugar creation will also be at the leaf only. This challenging problem of reasoning with entities evolving through time in a changing world is highly prevalent in text about natural biological processes, demonstrated by the paragraph in 1.1. Understanding what is happening in such texts is crucial for certain downstream tasks, such as procedure execution and validation, effect prediction, etc. On the other hand, this is challenging too because the entity state is continuously evolving over time, and the causal effects of actions on that state are most often implicit.

In order to try addressing this challenging domain of reading comprehension problem, researchers have constructed many synthetic datasets. One such dataset, namely, the bAbI dataset (Weston et al., 2015) includes queries about objects moved throughout a story paragraph, using machine-generated language over a specific domain with a very small lexicon base. The SCoNE dataset (Long et al., 2016 [5]) contains paragraphs describing a changing world state in form of three synthetic, deterministic domains, and assumes that the initial world state is explicitly known for each task. However, models developed for synthetic datasets often fail to handle the underlying complexity in natural language when applied to organic, real-world data (Hermann et al. [6], 2015; Winograd, 1972 [7]).

		Participants:				
Paragraph (seq. of steps):		**water**	**light**	**CO2**	**mixture**	**sugar**
	state0	**soil**	**sun**	**?**	**-**	**-**
Roots absorb water from soil						
	state1	**roots**	**sun**	**?**	**-**	**-**
The water flows to the leaf.						
	state2	**leaf**	**sun**	**?**	**-**	**-**
Light from the sun and CO2 enter the leaf.						
	state3	**leaf**	**leaf**	**leaf**	**-**	**-**
The light, water, and CO2 combine into a mixture.						
	state4	**-**	**-**	**-**	**leaf**	**-**
Mixture forms sugar.						
	state5	**-**	**-**	**-**	**-**	**leaf**

Time

Figure 1.2: *An annotated paragraph taken from ProPara dataset. The row indicates the existence as well as the location information of participants at each time step in the process (− means does not exist and ? means unknown location,). For example, the initial state (location) of the entity "water" is "soil".*

1.1.1 Dataset and Task

In this project, we work with a recently created dataset,namely ProPara (for **Pro**cess **Para**graphs), having 488 human-authored paragraphs of procedural text, along with 81,000 annotations about existence and location of entities in those paragraphs, with the ultimate task of predicting location and existence changes that take place across the process paragraph. This is the first ever dataset containing annotated, natural text data for real-world natural processes, along with a simple representation of the states of a pre-defined set of entities during those processes. A simple example is shown in Figure 1.2.

sentences	Questions	# domains	Vocab # words	# sentences	# unique sents	Avg words/sent
Natural	templated	183	2501	3.3k	3.2k	9.0

Table 1.1: ProPara Details

This dataset focuses upon a specific genre of procedural text data, i.e. simple scientific processes (e.g., photosynthesis, hydro-electricity production). A system that comprehends a process paragraph should be able to answer basic questions such as: What are the inputs to the process?, What conversions occur in the process and where?" Many of these questions reduce to understanding the basic dynamics of entities participating in the process, and we use this as our evaluation task: Given a process paragraph P and an entity e mentioned in P, predict:

Task 1 (Fine-grained/sentence-level evaluation)

(1) Is e created (destroyed, moved) in the process?
(2) When (step no.) is e created (destroyed, moved)?
(3) Where is e created (destroyed, moved from/to)?
We ask the above 3 categories of questions to the system-predicted table as well as the human-annotated table and thus calculate the F-1 score for each question category. These 3 category F-1 scores are aggregated using micro-avg and macro-avg and we report them as well.

Task 2 (Coarse-grained/document-level evaluation)

(1) What are the inputs (entities that existed before process but got destroyed during process) to the process?
(2) What are the outputs (entities that didn't exist before process but got created during process) of the process?
(3) What conversions occur, when and where?
(4) What movements occur, when and where?

We calculate the precision, recall and F-1 score for each category and aggregate them using averaging and report average precision, average recall and average F-1 score for task 2.

1.2 Our Contributions

- Proposed our model called ProGCN, which makes use of Graph Convolutional Network (GCN) on Semantic Role Labelling Graph. In this graph, the nodes are nothing but the paragraph token embeddings and the edges are the semantic role labels between these nodes. We observe that our approach beats many existing baselines on overall F-1 Score as well as significantly beats them on category 3 of questions which is about tracking the physical location of the entity.
- An exhaustive literature survey on models developed for this problem so far as this has remained a crucial and yet an underexplored problem and now the community working on this domain is expanding rapidly.

1.3 Related Work

1. **ProGlobal** - Dalvi et al. [8] propose the first approach to this problem by modelling the paragraph as a sequence of tokens as well as a sequence of time-steps (sentences). Think of each sentence as a basic unit of process in which some entity/entities participate and some state changes happen to it/them. They incorporate positional embeddings of the tokens in paragraph with respect to the mention of the entity under consideration in the current sentence. They also indicate the position of each sentence w.r.t. the current sentence (time-step) under consideration using the indicators 'C'(for current), 'F'(for following) and 'P'(for previous). This way, the entity and time-step information are incorporated using the position embeddings. On top of this, they apply a Bi-LSTM encoder for learning the token-level contextual embeddings. Now, they do the following on top of these paragraph embeddings -
    - Apply a linear + softmax layer to predict the state change of the entity into one of the 3 classes - (a) entity gets destroyed, (b) entity exists but location is not known or (c) entity exists and the location is known after the current time-step.
    - Use LSTM + softmax layer for learning a probability distribution over the words in the paragraph for start as well as end token of the location span in the paragraph. Hence, it is assumed that the state (physical location) of the entity after the current time-step is a span present in the paragraph. For e.g. in sentence, "water flows from river to ocean.", the state (location) of water after this step is the word 'ocean' and it is indeed a span in the paragraph.

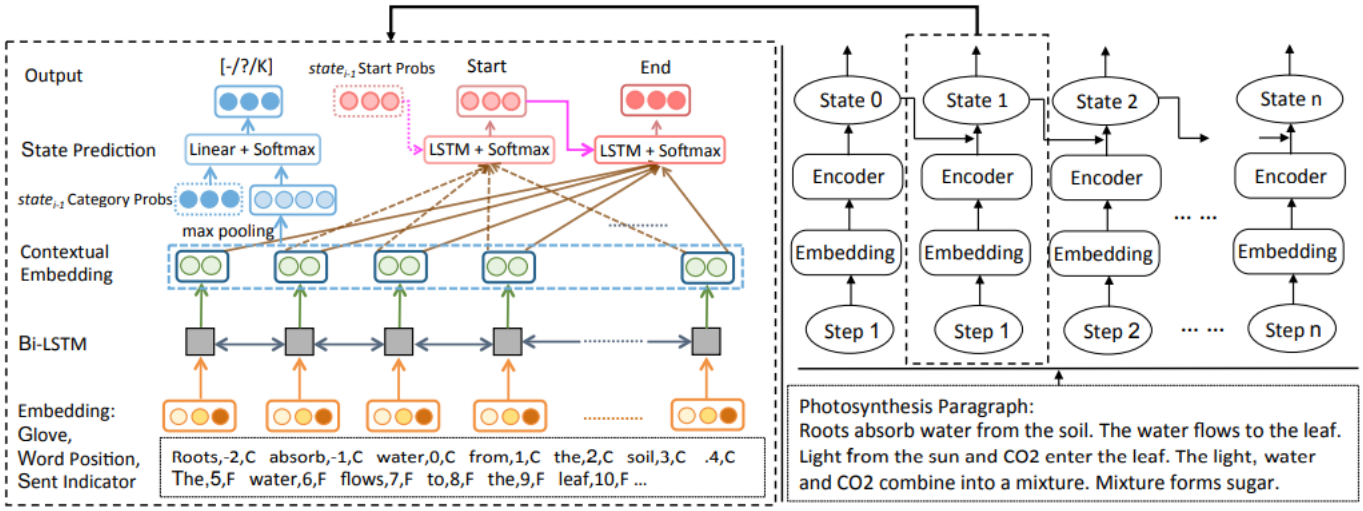


Figure 1.3: *ProGlobal*

2. **ProLocal** [8] - The main difference between ProLocal and ProGlobal is that ProLocal takes only the entity and the current sentence as the input and not the entire paragraph unlike ProGlobal. Moreover, they incorporate the POS tag information of the tokens using indicator variable whether the token is a verb or not. Even we successfully derive this idea for our own model described later.

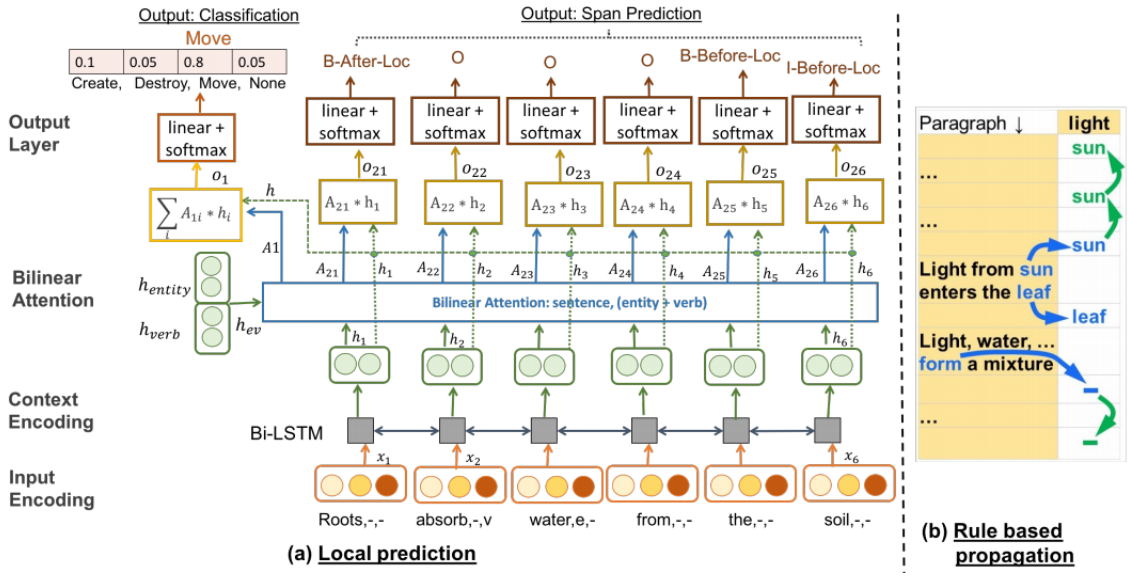
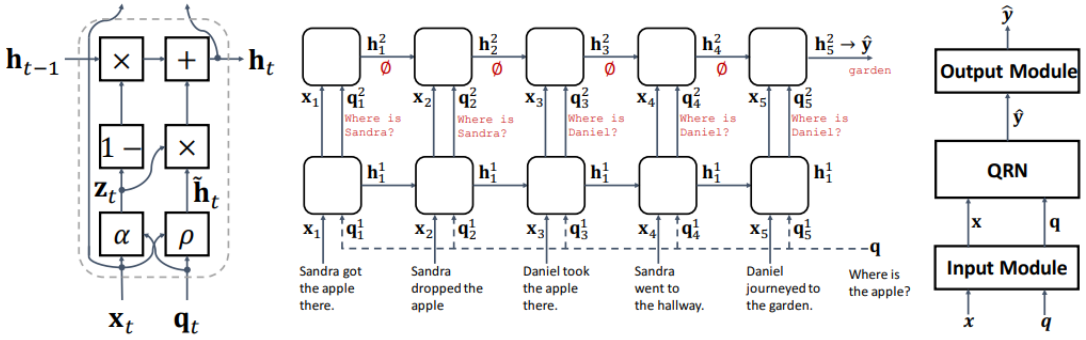


Figure 1.4: *ProLocal*



(a) QRN unit (b) 2-layer QRN (c) Overview

Figure 1.5: *QRN*

3. Memory Networks - Recently, many neural models have been developed to answer questions about the state of the world (entities) after a process, inspired in part by the bAbI dataset. Motivated by the general Memory Network architecture (Weston et al., 2014 [9]) and gated recurrent models such as GRU (Cho et al., 2014 [10]), **Recurrent Entity Networks (EntNet)** (Henaff et al., 2016 [11]) is a state-of-the-art method for bAbI story dataset. EntNet makes use of a dynamic memory of hidden states (memory blocks) to maintain a representation of the state of the world, making a gated update at each time step. Memory keys can be preset ("tied") to specific entities participating in the text, to encourage the memories to record information particularly about those entities. Similarly, **Query Reduction Networks (QRN)** (Seo et al., 2017b [12]) tracks the world state in a paragraph, represented as a hidden vector h. QRN does a gated propagation of h across each time step (corresponding to a state update), and uses h to modify (reduce) the query to keep pointing to the answer at each step (e.g., Where is the apple? at step 1 will be reduced to Where is Joe? at step 2 if Joe picks up the apple at step 1) (Figure 1.5).
   Both EntNet and QRN predict a final answer by calculating the probability distribution over the vocabulary using softmax layer.

4. **ProStruct (Tandon et al. [13])** - The crux of this paper is that it incorporates common sense knowledge in the model in 2 ways -

   - *Hard constraints*: These are the axioms of physics that they enforce while training the model itself unlike in ProLocal in which they incorporate these axioms only during inference using beam search. E.g., if an entity has been destroyed in an earlier step, then it cannot move/get destroyed in current step. An already existing entity cannot be created, etc.
   - *Soft constraints*: Soft constraints are incorporated by defining the prior probabilities P(π_j | e_j, topic) that entity e_j undergoes state change π_j in a sentence of text about a topic. These

priors are derived using Naive Bayes method on corpus obtained after Bing search for the topic under consideration. This prior probability is added to the log likelihood function to be optimized.

These constraints steer away the prediction of state change of an entity from an unlikely possibility. For example in Figure 1.7, decoder rules out the possibility of water getting created again in step 3 as it is created already in step 2 (Hard constraint). Also, the neural encoder predicts that the turbine moves in step 2, which is ruled out by decoder using soft constraint (turbine is a fixed entity and cannot change it's location).

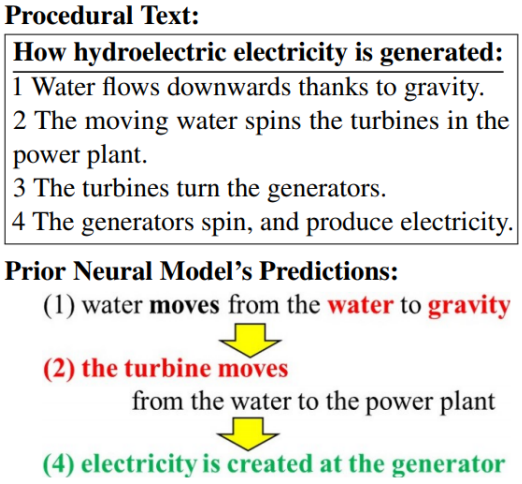


Figure 1.6: *Bad predictions (indicated in red) made by an existing neural model (ProGlobal) applied to a paragraph from the ProPara. ProGlobal predicts the entity state (location) at each sentence, but the implied movements violate commonsense constraints (e.g., 1. an object cannot move from itself to somewhere) and external knowledge-based common sense (e.g., 2. A turbine is a fixed entity which cannot move from one location to another)*

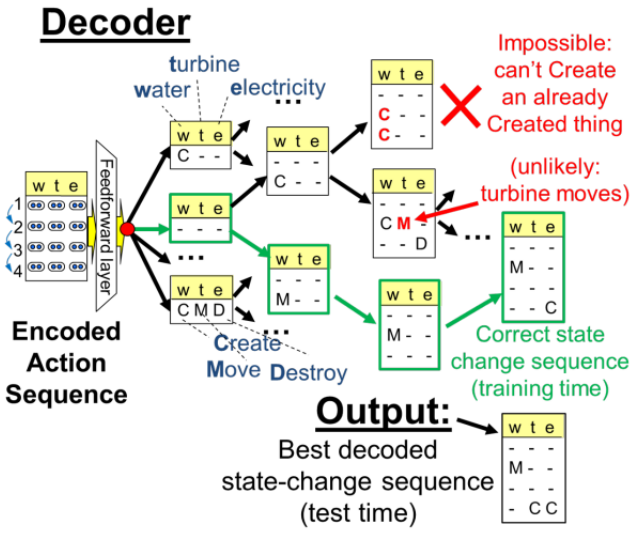


Figure 1.7: *ProStruct. The decoder rules out the possibility of water getting created again at step 3 when it is already existing. Also it rules out the possibility of the state change 'MOVE' for turbine at step 2 using external prior probability of state change 'NONE' to be high for turbine (turbine is a fixed entity which doesn't change its location). By narrowing down the search space in this way, the ProSturct does a beam search for decoding the best trajectory (one with maximum likelihood), given these commonsense constraints*

5. **KG-MRC (Das et al. [14])**: This is the most recent ICLR 2018 state-of-the-art paper on Propara dataset. They use the idea of *dynamic knowledge graph* which is nothing but the system predicted states (locations) of all entities from previous time steps. These entity and location representations are vector representations that get updated at each time step in a gated fashion. At each time t, the input to the MRC model is paragraph, query for location of entity e and dynamic knowledge graph at time t-1 as shown in Figure 1.8. The major highlights of this paper are as follows -

    - Performs soft co-reference of the location spans across sentences in paragraph using attention mechanism.
    - Keeps on updating the embeddings for entity at each time step such that the entities undergoing same state changes will be close in that embedding space. Hence, it models the interaction between the entities.

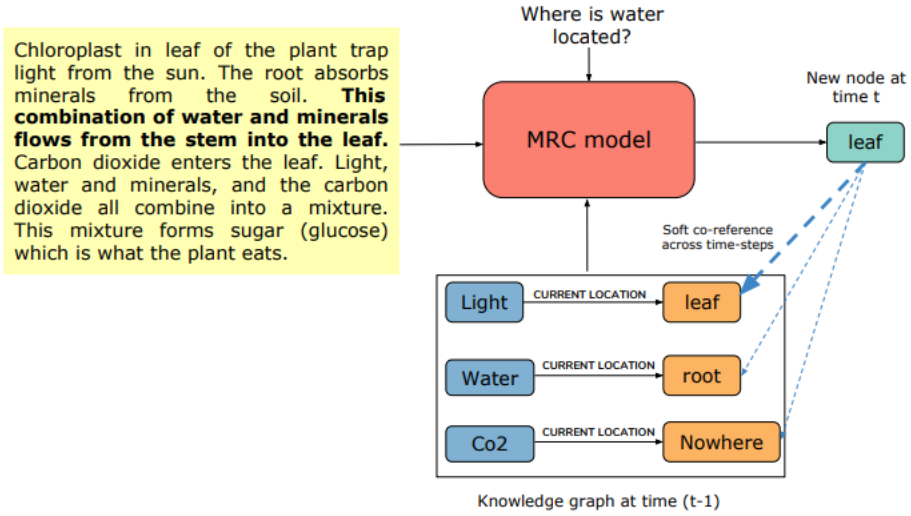


Figure 1.8: KG-MRC Model. The sentence at the current time step is highlighted. When the MRC model predicts a span (leaf) present in the graph at the previous time step, KG-MRC does soft attention and a gated update to preserve information across time steps. The thicker arrow shows higher attention weight between the old and new node.

Chapter 2

Question Answering on Procedural Documents - Proposed Methodology

2.1 Background

2.1.1 Graph Convolutional Networks (GCN)

GCNs are a generalization of CNN (Convolutional Neural Network) over graphs. GCN was introduced by Bruna et al.([15]) and later extended by Defferrard et al.([16]) with efficient localized filter approximation in the spectral domain. Kipf et al.([17]) propose a first-order approximation of localized filters through layerwise propagation rule. GCNs over syntactic dependency parse trees have recently been exploited in the field of Semantic Role Labeling [18], Neural Machine translation [19], neural document dating [20]. In our work, we try to use GCNs over Semantic Role Labeling Graph and prove its success for certain aspects. This idea is inspired by Neural Dater Model (Vashishth et al. [20]) for document classification problem.

2.1.2 Graph Convolution Network(GCN) on Undirected Graph

Let G = (V, E) be an un-directed graph, where V is set of n vertices and E the set of edges. The feature matrix $X \in R^{m \times n}$, whose rows are m-dimensional representation of input node u, $x_u \in R^m$, $\forall u \in V$. The hidden representation of output $h_v \in R^d$ of node v after a single layer of graph convolution can be obtained by considering only the immediate (1-hop) neighbours of v. This can be expressed as:

$$h_v = f(\sum_{(u \in N(v))}(Wx_u + b)), \quad \forall v \in V$$

Here, the model parameters $W \in R^{d \times m}$ and $b \in R^d$ are learnt in a task-oriented setting using first - order gradient optimization. Note that $N(v)$ refers to the set of neighbours of v and f is a non-linear

activation function. We use ReLU activation function for our task (ReLU(x) = max(0,x)).
In order to aggregate information from multi-hop neighbours, multiple layers of GCN can be stacked one on top of another, for e.g. 2 layers of GCN means aggregating information of 2-hop neighbours of a node . In fact, h_v^{k+1}, hidden representation of node v after k^{th} GCN layer can be expressed as:

$$h_v^{k+1} = f(\sum_{(u \in N(v))}(W^k h_u^k + b^k)), \quad \forall v \in V \tag{2.1}$$

where h_u^k is the input to the k^{th} GCN layer or output of (k-1)th GCN layer.

2.1.3 GCN on Labeled and Directed Graph

In this section, we consider the case of applying GCN over graphs where each edge is directed as well as labeled. In this setting, let the edge from node u to v with label *l(u, v)* be denoted as *(u, v, l(u, v))*. While a few recent works focus upon GCN over directed graphs ([21], [18]), none of them consider the edge labels into account. We handle both direction as well as label by incorporating label and direction specific filters.
Our hypothesis is that the information in a directed graph along an edge need not propagate only along its direction. Following this [18] we define an updated edge set $\mathcal{E}'$ which expands the original set $\mathcal{E}$ by incorporating inverse as well as self-loop edges.

$$\mathcal{E}' = \mathcal{E} \bigcup \{(v, u, l(u, v)^{-1}) \mid (u, v, l(u, v)) \in \mathcal{E}\} \bigcup \{(u, u, \Upsilon) \mid u \in V\} \tag{2.2}$$

Here, $l(u, v)^{-1}$ denotes the inverse edge label corresponding to label l(u,v), and Υ is a special empty relation symbol denoting self-loop edges. We now define h_v^{k+1} as the nbedding for vertexode em v after k^{th} GCN layer applied over the directed and labeled graph as:

$$h_v^{k+1} = f(\sum_{(u \in N(v))}(W_{l(u,v)}^k h_u^k + b_{l(u,v)}^k)). \tag{2.3}$$

Note that, in this case, the parameters $W_{l(u,v)}^k$ and $b_{l(u,v)}^k$ are specific to an edge label.

2.1.4 Incorporating Edge Importance

In most practical settings, we would not like to give equal importance to all the edges. Label specific edge-wise gating may be used in a GCN to give importance to relevant edges and subdue the noisy ones.([20], [22], [18]) used gating for similar reasons and also obtained high performance gain. For the k^{th} layer, we compute gating value for a given labeled edge *(u, v, l(u, v))* given by:

$$g^k_{l(u,v)} = \sigma(h^k_u.w^k_{l(u,v)} + b^k_{l(u,v)})$$

where, $\sigma(.)$ is the sigmoid function and $w^k_{l(u,v)}$ and $b^k_{l(u,v)}$ are label specific gating parameters. Thus, gating helps in making the model robust to the noisy and erroneous labels and directions of the input graphs. While incorporating edge gating, the GCN node representation may be computed as follows.

$$h^{k+1}_v = f(\sum_{(u \in N(v))} \; g^k_{l(u,v)} \times (W^k_{l(u,v)} h^k_u + b^k_{l(u,v)})). \tag{2.4}$$

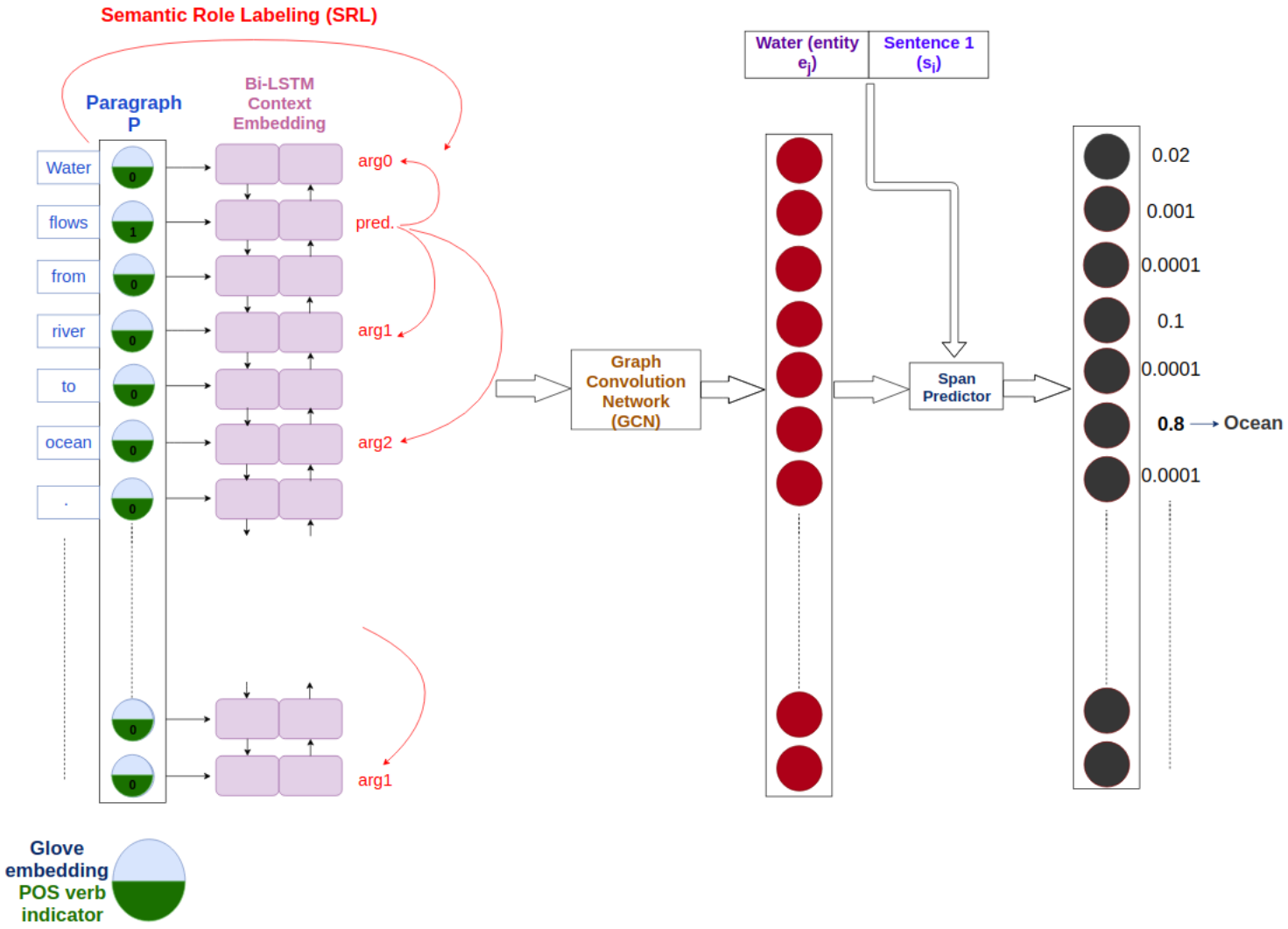


Figure 2.1: ProGCN Model. The representations of entity (e_j) and sentence (s_i) are described in Figure 2.2 and Figure 2.3 respectively.

2.2 ProGCN Details

The entity state tracking problem can be considered as of a span prediction problem ([23], [24]) in which the model needs to learn 2 probability distributions over the words in the paragraph. The first one is for the start token of the span and the second one is for end token. In this section, we give an overview of ProGCN as shown in Figure 2.1.

ProGCN is a deep learning-based span prediction system. It takes in input as a procedural document along with a fixed set of entities E = $\{e_1, e_2, ..., e_{|E|}\}$ participating in that procedural document. It outputs a table in which the columns correspond to entities and rows correspond to sentences. Each sentence can be thought of as a time-step (basic unit) of the process. The $(i, j)^{th}$ entry of the table is the state (physical location) of entity e_j after time step i (sentence s_i). This entry is a span in the document and thus the hypothesis is that the state of the entity after every time step (in case the entity exists) is essentially a substring of the paragraph.

ProGCN model consists of many components as described in detail below : –

2.2.1 Paragraph Context embedding (Bi-LSTM)

Consider a paragraph P with n words (tokens) $w_1, w_2, ..., w_n$. We represent each word by a k-dimensional pre-trained word embedding concatenated with a binary POS indicator for that token indicating whether the token is a verb or not (1 if it is a verb, otherwise 0). For the experiments in this model, we use GloVe [25] embeddings and for POS tagging, we use Stanford CoreNLP POS tagger [26]. Let us stack together these token-level embeddings to obtain the document representation X $\in R^{n\times(k+1)}$. After this, we apply Bi-directional LSTM (Bi-LSTM) [27] to encode the input matrix X and obtain contextual embedding for each word. After stacking contextual embeddings of all these words in the paragraph, we obtain the new document representation matrix $H^{cntx} \in R^{n\times r_{cntx}}$. Note that in this representation, each token is represented by a r_{cntx}-dimensional vector.

2.2.2 entity representation

We consider all the occurences of entity e_j in pargraph P using simple string matching and aggregate their contextual embeddings (obtained above) using simple averaging. This gives us the entity embedding, denoted by e_j (Figure 2.2).

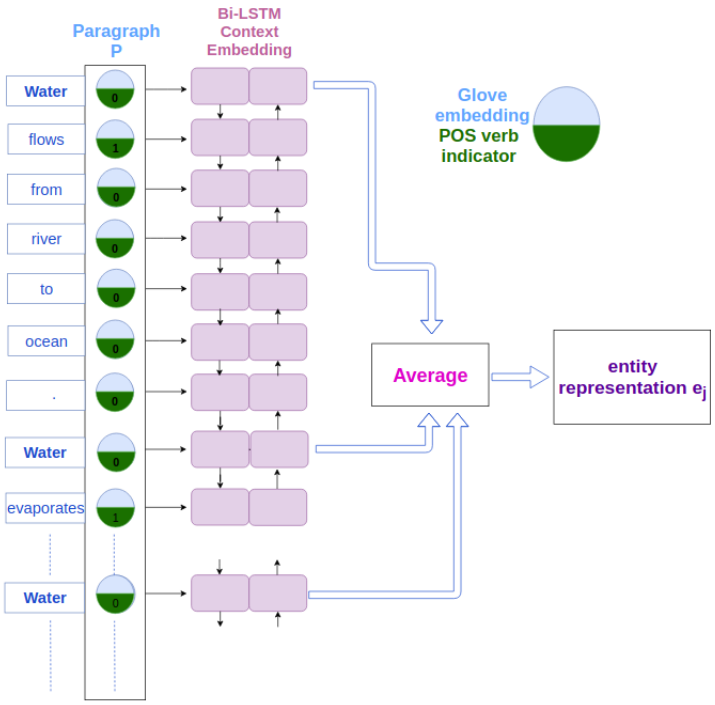


Figure 2.2: Representation for entity 'water'(e_j)

2.2.3 Sentence context embedding (Bi-LSTM)

To incorporate time-step information at time step i, we take the sentence s_i and learn the context embeddings for this the same way we did for the paragraph P as above. After this, we calculate the attention score of each word in the sentence w.r.t. entity e_i and aggregate these word embeddings of sentence using these attention scores.

$$\alpha_{ik} = \frac{exp(score(h_k^{(i)}, e_j))}{\sum_k exp(score(h_k^{(i)}, e_j))}$$

where $h_k^{(i)}$ is the contextual representation of k^{th} word in i^{th} sentence obtained after applying Bi-LSTM to the sentence, score function is the dot product between 2 vectors.
The final sentence (time-step) representation is given by -

$$s_i = \sum_k \alpha_{ik} \times h_k^{(i)}$$

Note that s_i is the entity-centric sentence representation for time step i (Figure 2.3). We concatenate

e_j and s_i and preserve this vector for the final layer of node classification discussed below.

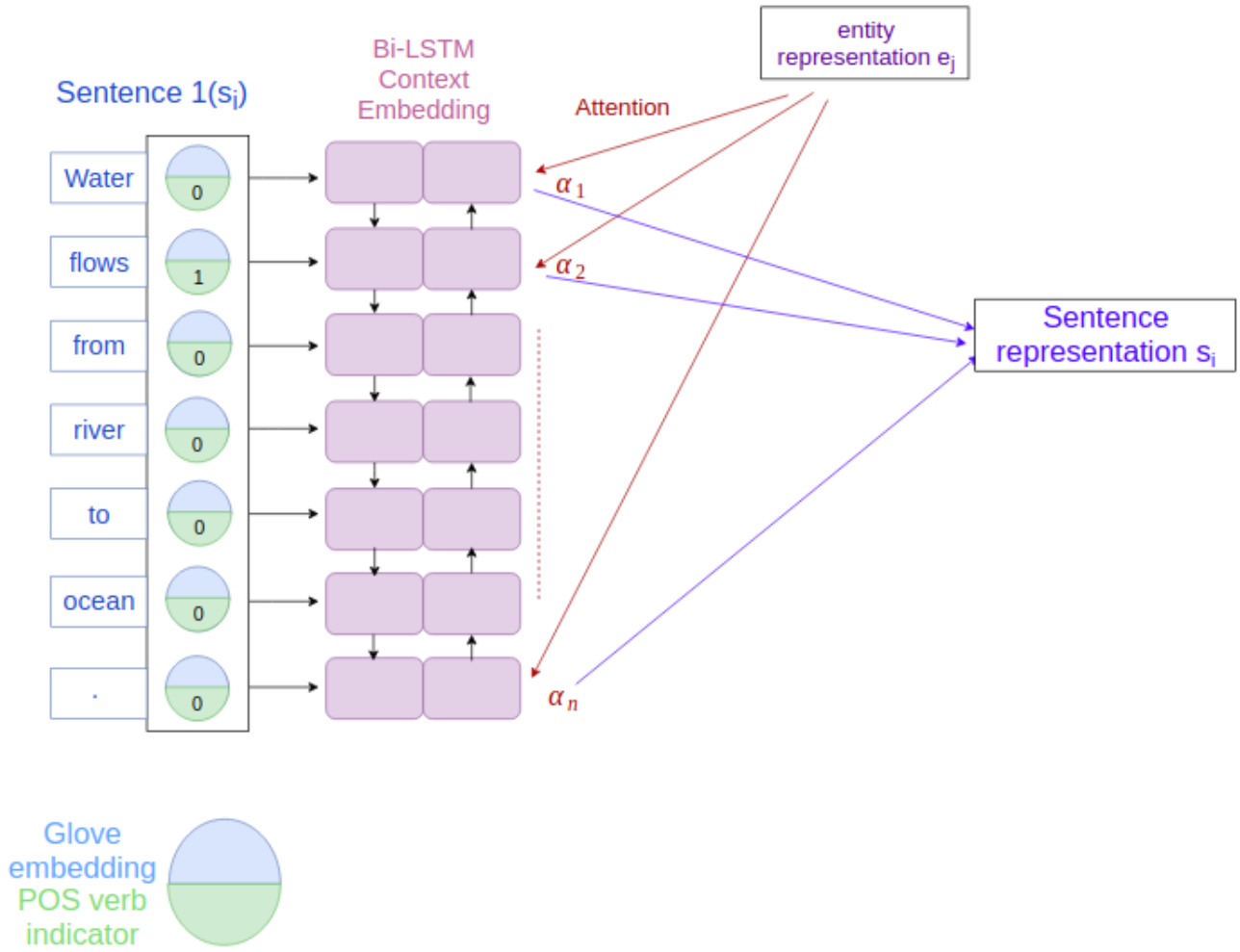


Figure 2.3: Representation for sentence 1(s_i)

2.2.4 Semantic Embedding (SRL - GCN)

Although Bi-LSTM is quite efficient in capturing immediate local context of a word in paragraph, it may not be as efficient in capturing longer range dependencies among words in the paragraph. For example, in the sentence "The bacteria, which lives in the marshy area, gets buried in the sediment.", we would want the embedding of token 'sediment' to be directly affected by "bacteria", the state (physical location) output by the model should be the token "sediment". A Semantic Role Labeling Graph may be useful in capturing such longer-range dependencies. In fact, corresponding to the predicate "buried", the arguments are - "The bacteria, which lives in marshy area"(arg0) and "sediment"(arg1). Clearly the expected output i.e. the token "sediment" is nothing but arg1 corresponding to the arg0 which refers

to the bacteria w.r.t. the predicate "buried". This is the main motivation behind using Semantic Role Labeling (SRL) by using a GCN run over the SRL graph of the document. We describe this in detail below.

The contextual embedding, $H^{cntx} \in R^{n \times r_{cntx}}$ obtained in the previous step is used as input to this layer. For a given paragraph (document), we first extract its Semantic Role labeling graph by applying the pre-trained SRL model (He et al., (2018) + ELMO [28]) [29] on each sentence in the document individually. We now apply the Graph Convolution Network (GCN) over this SRL graph using the formulation of GCN as presented in subsection . We call this GCN the Semantic GCN or SRL-GCN.

Since SRL-GCN operates over the SRL graph and uses equation 2.4 for learning the embeddings, the number of parameters in SRL-GCN is going to be directly proportional to the number of SRL edge types. Our pre-trained SRL model returns 67 different SRL edge types. This is a large number and hence is going to over-parameterize SRL-GCN significantly, thereby leading to the possibility of high model variance (overfitting). In order to mitigate this issue, we use only three edge types in SRL-GCN. For each edge connecting nodes w_i and w_j in $\mathcal{E}'$ (see Equation 2.2), we determine its new type $L(w_i, w_j)$ as follows:

- $L(w_i, w_j) = \rightarrow$ if $(w_i, w_j, l(w_i, w_j)) \in \mathcal{E}'$, i.e., if the edge is an original SRL edge
- $L(w_i, w_j) = \leftarrow$ if $(w_i, w_j, l(w_i, w_j)^{-1}) \in \mathcal{E}'$, i.e., if the edge is an inverse edge
- $L(w_i, w_j) = \Upsilon$ if $(w_i, w_j, \Upsilon) \in \mathcal{E}'$, i.e., if the edge is a self-loop with $w_i = w_j$

SRL-GCN now estimates embeddings $h^{sem}_{w_i} \in R^{r_{sem}}$ for each token w_i in the paragraph using the formulation shown below.

$$h^{sem}_{w_i} = f(\sum_{(w_j \in N(w_i))} \quad g_{L(w_i,w_j)} \times (W_{L(w_i,w_j)} h^{cntx}_{w_j} + b_{L(w_i,w_j)})).$$

Please note SRL-GCN's use of the new edge types $L(w_i, w_j)$ above, instead of the $l(w_i, w_j)$ types used in Equation 2.4. By stacking embeddings for all the tokens together, we get the new embedding matrix $H^{sem} \in R^{n \times r_{sem}}$ representing the document (paragraph).

2.2.5 Span Predictor

Consider the entity representation e_j and sentence representation s_i obtained from subsections 2.2.2 and 2.2.3 respectively. We concatenate these 2 vectors and obtain a single vector

$$es_{ji} = [e_j; s_i]$$

Note that the dimension of es_{ji} is $2r_{cntx}$.
We project this down to r_{sem} using a linear projection matrix of size $2r_{cntx} \times r_{sem}$.

$$es_{ji}^{sem} = es_{ji} \times W^{2r_{cntx} \times r_{sem}}$$

This projection is done in order to calculate the dot product (attention score) between the vector es_{ji}^{sem} and each word of the paragraph having dimension r_{sem} obtained after applying GCN in subsection 2.1.4.

$$\beta_{(k)}^{sem} = \frac{exp(score(h_{w_k}^{sem}, es_{ji}))}{\sum_k exp(score(h_{w_k}^{sem}, es_{ji}))}$$

Intuitively, $\beta_{(k)}^{sem}$ represents how relevant is the k^{th} word in the paragraph w.r.t. entity e_j and sentence s_i. We multiply these attention scores with the respective embeddings in the paragraph and obtain new embedding for each token given by -

$$h_{w_k}^{final} = \beta_{(k)}^{sem} \times h_{w_k}^{sem}$$

By stacking embeddings for all the tokens together, we get the new embedding matrix $H^{final} \in R^{n \times r_{sem}}$ representing the document.
Since this final layer document representation is corresponding to entity e_j and time step i (sentence s_i), let me call this representation as $H_{(i)}^{final}$ i.e. the representation of paragraph w.r.t. entity e_j at time step i.
Finally, we concatenate $H_{(i)}^{final}$ with $H_{(i-1)}^{final}$ and pass it through a single layer neural network and softmax layer to learn a probability distribution over all the tokens of paragraph. This way, we learn 2 different probability distributions - first for the start token of the span and second for end token. We consider the start and end tokens corresponding to maximum probability values of these 2 distributions and output the span with these as start and end tokens as our predicted state (physical location) of entity e_j after time step i.

$$H_{(i)}^{final} = [H_{(i)}^{final}; H_{(i-1)}^{final}]$$

Note that $H_{(i)}^{final} \in R^{n \times 2r_{sem}}$.
Now we apply three classifiers on top of $H_{(i)}^{final}$ as follows :

- State change prediction: We first aggregate the $H_{(i)}^{final}$ along first dimension by summing the embeddings of individual tokens, let me call it 'reduce_sum' operation. Then we apply a dense layer with linear activation function to project it into 3-dimensional space and finally apply softmax function to learn probability distribution over 3 state change categories - (a) entity gets destroyed, (b) entity exists but location is not known or (c) entity exists and the location is known, after the

current time-step. We select the category with largest probability value. Also, we define 'category_loss' as cross entropy loss between gold category (one-hot vector of size 3) and the probability distribution learnt above.

$$H_{(i)}^{final^{(1)}} = reduce\_sum(H_{(i)}^{final}, axis = 0) \times W_c^{2r_{sem} \times 3}$$

$H_{(i)}^{final^{(1)}} \in R^{1 \times 3}$.

$$state\_prob[k] = \frac{exp(H_{(i)}^{final^{(1)}}[k])}{\sum_k exp(H_{(i)}^{final^{(1)}}[k])}, \qquad k \in \{1, 2, 3\}$$

$$\text{category\_loss} = \text{cross\_entropy(state\_prob, state\_gold)}$$

- Span prediction: We learn 2 probability distribution - one for the start of the span and other for the end token of span. First, we apply 2 dense layers to project $H_{(i)}^{final}$ from $R^{n \times 2cntx}$ to $R^{n \times 1}$. Then we apply softmax function on these n-dimensional vectors to learn probability distributions for start and end token respectively. The predicted span is the one from token i to token i' such that $i \leq i' \leq i$+max_len and $start\_prob[i] \times end\_prob[i']$ is maximised (Chen et al., 2017 [24]). We use max_len value as 5 for our task. We define start_loss and end_loss as cross-entropy between gold start token (one-hot vector of size n) and start_prob and between gold end token (one-hot vector of size n) and end_prob respectively.

$$H_{(i)}^{final^{(2)}} = H_{(i)}^{final} \times W_s^{2r_{sem} \times 1}$$

$$H_{(i)}^{final^{(3)}} = H_{(i)}^{final} \times W_e^{2r_{sem} \times 1}$$

$H_{(i)}^{final^{(2)}}, H_{(i)}^{final^{(3)}} \in R^{n \times 1}$

$$start\_prob[k] = \frac{exp(H_{(i)}^{final^{(2)}}[k])}{\sum_k exp(H_{(i)}^{final^{(2)}}[k])}, \qquad k \in \{1, 2, ..., n\}$$

$$end\_prob[k] = \frac{exp(H_{(i)}^{final^{(3)}}[k])}{\sum_k exp(H_{(i)}^{final^{(3)}}[k])}, \qquad k \in \{1, 2, ..., n\}$$

$$\text{start\_loss} = \text{cross\_entropy(start\_prob, start\_gold)}$$

$$\text{end\_loss} = \text{cross\_entropy(end\_prob, end\_gold)}$$

Finally, our loss function to be optimized is given by accumulating the 3 loss functions defined above -

$$\textbf{Loss} = \text{category\_loss} + \text{start\_loss} + \text{end\_loss}$$

2.3 Experimental Setup

Dataset and Evaluation Metric - Described in subsection 1.1.1

Baselines - Described in section 1.3

Hyperparameters - By default, edge gating (2.1.4) is used in all GCNs for ProGCN. We use 300-dimensional GloVe embeddings and 256-dimensional hidden state for both Bi-LSTM as well as GCN with dropout 0.1. We used Adam [30] with 0.0001 learning rate for training. We used single layer of SRL-GCN for our model. All these hyperparameters are obtained after tuning on validation set.

2.4 Results

2.4.1 Ablation Comparisons

In this section, we demonstrate the efficacy of using SRL-GCN and POS tagging for the problem. We evaluate different variants of ProGCN on the ProPara dataset. Especially, we validate the significance of using SRL-GCN for boosting the F-1 score for category 3 of questions. This category is about querying the physical location of an entity at each time step, and is a challenging category for which even upper human bound is 63%. Table 2.1 shows the ablation study of our model.

Question type (# Questions)	Original(ProGCN)	- POS tagging	- SRL-GCN
Cat-1 (750)	55.15	54.79	51.83
Cat-2 (601)	10.99	7.35	8.45
Cat-3 (823)	26.29	24.57	16.32
macro-avg	30.81	28.9	25.53
micro-avg	32.02	30.24	26.39

Table 2.1: Ablation Comparisons (Task 1 (1.1.1))

2.5 Ablation Analysis

- Removing POS verb indicator embedding marginally deteriorates the performance (around F-1 score of 2).

- Removing SRL-GCN deteriorates the performance significantly, especially for category 3 of questions (F-1 score of around 10). This validates the effectiveness of GCN for tracking physical location of entity and hence it's significance for category 3.
- Another observation (not mentioned in table) is that using multiple GCN layers significantly deteriorates the overall performance (around 7 F1-score). This observation is consistent with [18], since multiple layers of GCN become redundant when applied on top of Bi-LSTM. We also observe that removing edge gating from our model deteriorates its overall performance.

2.6 Performance Analysis

In order to evaluate our model, we compare it against the existing models (1.3). The final results are summarized in tables 2.2 and 2.3.

Question type(# Questions)	QRN [12]	EntNet [11]	SRL Rule-based [31]	ProLocal [8]	ProGlobal [8]	KG-MRC [14]	**ProGCN (Ours)**	Human Upper Bound
Cat-1 (750)	52.37	51.62	57.14	62.65	**62.95**	62.86	55.15	91.67
Cat-2 (601)	15.51	18.83	20.33	30.50	36.39	**40.00**	10.99	87.66
Cat-3 (823)	10.92	7.77	2.4	10.35	**35.9**	**38.23**	**26.29**	**62.96**
macro-avg	26.26	26.07	26.62	34.50	45.08	**47.03**	30.81	80.76
micro-avg	26.49	25.96	26.24	33.96	45.37	**46.62**	32.02	79.69

Table 2.2: Task 1 (1.1.1) F-1 Score

Models	Precision	Recall	F-1
ProLocal	**77.4**	22.9	35.3
QRN	55.5	31.3	40.0
EntNet	50.2	33.5	40.2
ProGCN (Ours)	71.5	34.4	46.5
ProGlobal	46.7	**52.4**	49.4
ProStruct	74.2	42.1	53.75
KG-MRC	64.52	50.68	**56.77**

Table 2.3: Task 2 (1.1.1)

- Our Model performs quite well on Task 1 category 3 of questions, which is about querying the location of entity at each time step. It outperforms even ProLocal model by around 16 F-1 score. The human upper bound is also around 63% for this intrinsically challenging task.
- We compare our model against SRL rule-based model ([31]) and it remarkably outperforms theirs for category 3 by around 24 F-1 score. This observation validates our hypothesis that using side information (SRL, Dependency Parsing, POS tagging) as a part of neural structured prediction model may be much more efficient than creating a set of handcrafted rules out of it.
- The accuracy for our model on Task 1 category 1, which is about querying the state change of entity, is much better than random prediction (50%) and is comparable with other models as well.
- The overall performance of our model on task 2 is better as compared to that on task 1.
- The major limitation of our model is that it does not perform good for Task 1 category 2, which is about querying the time-step at which an entity gets created or gets destroyed or moves. Thus time step information is not embedded correctly in our model. In order to try solving this problem, we use the teacher-forcing technique described in detail in next section.

2.7 How Teacher Forcing could help in incorporating temporal information !!!

Teacher forcing ([32], [33]) is an efficient technique for training sequence-2-sequence models, that uses model output from a prior time step as an input at current time step. Thus during training time it makes use of gold labels (an efficient use of data) but while inference time (when we don't have access to gold labels), we use predicted label from previous time step as input for current time step.
For our problem, we have access to gold location labels of entities at each time step. Thus, instead of concatenating $H_{(i)}^{final}$ with $H_{(i-1)}^{final}]$ (span predictor (2.2.5)), we take the gold location (concatenation of start token position i and end token position i') and replicate it n times (one for each paragraph token), so it's dimensionality becomes $R^{n\times 2}$. Let me call this vector y_{i-1}. We concatenate this with $H_{(i)}^{final}$ so that :

$$H_{(i)}^{final} = [H_{(i)}^{final}; y_{i-1}]$$

Note that after concatenation, $H_{(i)}^{final} \in R^{n\times(r_{sem}+2)}$ On top of this vector $H_{(i)}^{final}$, we proceed in same way for rest of the span prediction as we did for ProGCN (2.2.5). Surprisingly, this approach enhances the accuracy for category 2 as stated in table 2.4.

Question type (# Questions)	Original(ProGCN)	ProGCN + teacher forcing
Cat-1 (750)	55.15	**59.36**
Cat-2 (601)	10.99	**18.9**
Cat-3 (823)	**26.29**	8.99
macro-avg	**30.81**	29.08
micro-avg	**32.02**	29.11

Table 2.4: ProGCN vs Teacher Forcing model

We observe that

- Teacher forcing helps in achieving better F-1 score for category 1 and 2. This means it helps in predicting state change of an entity as well as in querying for time step for entity participation better than ProGCN. Our hypothesis is that by giving previous time step gold output as input to current step, the model is somehow able to distinguish across various time steps in a much better way that before and hence querying time step information becomes more feasible.
- However, by applying teacher forcing, the category 3 performance deteriorates and hence the overall performance remains almost same.

2.8 Discussion

For our experiments, we also tried to use Transformer model [34] for encoding the paragraph (document) and sentence. The results with this approach were quite worse compared to using standard Bi-LSTM, mostly because of over-parametrization (high model variance due to large complexity). Moreover, for span prediction task, we tried to use sequence-to-segment [23] model, which is quite effective for video segmentation task, but again the results were not good probably because the model is not good at domain adaptation/transfer learning. One observation that is pertinent across various models is the trade-off between various categories of questions. This means that a model which is good at capturing time step information may not be good at predicting the location for an entity and vice versa. We need to come up with ways of combining the core aspects of various models. This combination needs to be more elegant than simply ensembling the outputs of these models (Model Ensembling), by treating them as black boxes.

Chapter 3

Conclusion and Future Work

3.1 Workflow Extraction ('Process Graphs')

Earlier methods such as ProRead(Berant et al. [35]) have tried to extract something called a 'Process Graph' out of prcoedural text in which nodes are the events as well as arguments of those events, while edges are the event-event temporal ordering and event-argument Semantic Role Labeling. Such event-ordering graphs have been proven to be useful for certain downstream tasks such as question-answering ([35])(Figure 3.1).

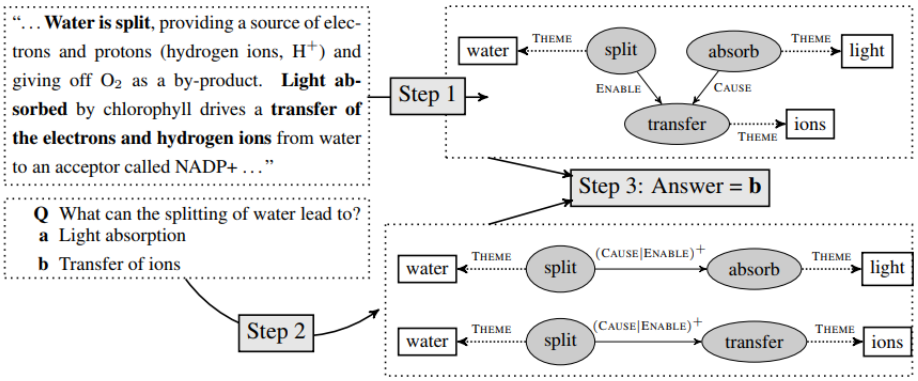


Figure 3.1:

We also extracted this 'process graph' out of Ericsson manuals but using CATENA (CAusal and TEmporal relation extraction from NAtu-ral language texts [36]) for temporal ordering of events in the text. Moreover, for event-argument labeling, we use pre-trained SRL model (He et al., (2018) + ELMO [28]) [29]. These are the state-of-the-art models for their respective tasks. We are currently

exploring how these pre-trained models can be applied to learn meaningful and easy-to-interpret representations of text that is useful for downstream tasks such as reasoning, automation etc. We would like to compare this representation of documents with something called the Resource Description Framework (RDFs) whose samples are provided by a separate team in Ericsson Research. If there exists some similarity between these 2 representations, then we can establish that the pre-trained models applied on these documents are helpful for downstream reasoning.

3.2 Entity Tracking

In our current, work we validate the effectiveness of Graph Convolution Networks (GCN) for tracking the physical location of entity after each time step and still a lot more needs to be explored in this direction.

As a future line of work, we would like to evaluate the effectiveness of existing models for tracking different entity states. While in our current line of work, the entity state of interest is the physical location, it could be shape/composition of ingredients in a food recipe, voltage of a power supply in a troubleshooting document, temperature of ingredients in a chemical recipe etc. This domain adaptation/transfer learning would require fine-tuning of existing models on newly created annotated datasets for tracking various properties of interest. It would be interesting to see how well do these existing models generalize to tracking any property of interest of participants in a text describing a process of any domain.

3.3 Combining these two

Consider a troubleshooting document describing how to troubleshoot a power supply exceeding a threshold voltage. For end-to-end automation, we would like to track voltage of power supply from console logs and if this exceeds the threshold, then the workflow/process graph on troubleshooting document for power supply could essentially order the subtasks to be performed for troubleshooting. This is something which we are currently working upon as a part of Ericsson Research project.

Bibliography

[1] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," *arXiv preprint arXiv:1611.01603* (2016).

[2] C. Clark and M. Gardner, "Simple and effective multi-paragraph reading comprehension," *arXiv preprint arXiv:1710.10723* (2017).

[3] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," *arXiv preprint arXiv:1707.07328* (2017).

[4] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250* (2016).

[5] R. Long, P. Pasupat, and P. Liang, "Simpler context-dependent logical forms via model projections," *arXiv preprint arXiv:1606.05378* (2016).

[6] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *Advances in neural information processing systems*, 1693–1701 (2015).

[7] T. Winograd, "Understanding natural language," *Cognitive psychology* **3**(1), 1–191 (1972).

[8] B. D. Mishra, L. Huang, N. Tandon, W.-t. Yih, and P. Clark, "Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension," *arXiv preprint arXiv:1805.06975* (2018).

[9] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *arXiv preprint arXiv:1410.3916* (2014).

[10] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259* (2014).

[11] M. Henaff, J. Weston, A. Szlam, A. Bordes, and Y. LeCun, "Tracking the world state with recurrent entity networks," *arXiv preprint arXiv:1612.03969* (2016).

[12] M. Seo, S. Min, A. Farhadi, and H. Hajishirzi, "Query-reduction networks for question answering," *arXiv preprint arXiv:1606.04582* (2016).

[13] N. Tandon, B. D. Mishra, J. Grus, W.-t. Yih, A. Bosselut, and P. Clark, "Reasoning about actions and state changes by injecting commonsense knowledge," *arXiv preprint arXiv:1808.10012* (2018).

[14] R. Das, T. Munkhdalai, X. Yuan, A. Trischler, and A. McCallum, "Building dynamic knowledge graphs from text using machine reading comprehension," *arXiv preprint arXiv:1810.05682* (2018).

[15] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203* (2013).

[16] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 3844–3852 (2016).

[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907* (2016).

[18] D. Marcheggiani and I. Titov, "Encoding sentences with graph convolutional networks for semantic role labeling," *arXiv preprint arXiv:1703.04826* (2017).

[19] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an, "Graph convolutional encoders for syntax-aware neural machine translation," *arXiv preprint arXiv:1704.04675* (2017).

[20] S. Vashishth, S. S. Dasgupta, S. N. Ray, and P. Talukdar, "Dating documents using graph convolution networks," *arXiv preprint arXiv:1902.00175* (2019).

[21] M. Yasunaga, R. Zhang, K. Meelu, A. Pareek, K. Srinivasan, and D. Radev, "Graph-based neural multi-document summarization," *arXiv preprint arXiv:1706.06681* (2017).

[22] T. H. Nguyen and R. Grishman, "Graph convolutional networks with argument-aware pooling for event detection," in *Thirty-Second AAAI Conference on Artificial Intelligence*, (2018).

[23] Z. Wei, B. Wang, M. H. Nguyen, J. Zhang, Z. Lin, X. Shen, R. Mech, and D. Samaras, "Sequence-to-segment networks for segment detection," in *Advances in Neural Information Processing Systems*, 3507–3516 (2018).

[24] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading wikipedia to answer open-domain questions," *arXiv preprint arXiv:1704.00051* (2017).

[25] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543 (2014).

[26] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 55–60 (2014).

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation* **9**(8), 1735–1780 (1997).

[28] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365* (2018).

[29] L. He, K. Lee, O. Levy, and L. Zettlemoyer, "Jointly predicting predicates and arguments in neural semantic role labeling," *arXiv preprint arXiv:1805.04787* (2018).

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

[31] P. Clark, B. Dalvi, and N. Tandon, "What happened? leveraging verbnet to predict the effects of actions in procedural text," *arXiv preprint arXiv:1804.05435* (2018).

[32] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation* **1**(2), 270–280 (1989).

[33] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, 1171–1179 (2015).

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 5998–6008 (2017).

[35] J. Berant, V. Srikumar, P.-C. Chen, A. Vander Linden, B. Harding, B. Huang, P. Clark, and C. D. Manning, "Modeling biological processes for reading comprehension," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1499–1510 (2014).

[36] P. Mirza and S. Tonelli, "Catena: Causal and temporal relation extraction from natural language texts," in *The 26th international conference on computational linguistics*, 64–75, ACL (2016).