CLaC-np at SemEval-2021 Task 8: Dependency DGCNN

Nihatha Lathiff, Pavel Khloponin, and Sabine Bergler
CLaC Labs, Concordia University
Montreal, Canada
{f_lathif, p_khlopo, bergler}@cse.concordia.ca

Abstract

MeasEval aims at identifying quantities along with the entities that are measured with additional properties within English scientific documents. The variety of styles used makes measurements, a most crucial aspect of scientific writing, challenging to extract. This paper presents ablation studies making the case for several preprocessing steps such as specialized tokenization rules. For linguistic structure, we encode dependency trees in a Deep Graph Convolution Network (DGCNN) for multi-task classification.

1 Introduction

Scientific articles contain many quantities which have to be linked to their measured entities. Identifying quantities may seem as simple as digit recognition, but numbers alone are not informative. The entities and properties being measured, while crucial information, are difficult to extract. SemEval 2021 Task 8 (Harper et al., 2021) is a semantic relation extraction task consisting of 5 subtasks: identifying *quantities* and their *modifying* attributes, identifying *measured entities* and their *properties* as well as *qualifying* attributes, if specified.

Recent reports on the strong performance of purely neural models for NLP tasks often underreport the data preprocessing and postprocessing steps that accompany them. Preprocessing significantly influences overall performance. Typical NLP preprocessing steps include sentence splitting and tokenization, sometimes followed by task relevant gazetteer annotation, possibly named entity recognition (NER), part-of-speech (POS) tagging and dependency parsing. These preprocessing steps are so common that many different packages perform them, such as Stanford CoreNLP (Manning et al., 2014), spaCy[1], and NLTK (Bird et al., 2009). Linguistically inspired features are, however, not regularly exploited and we present here one attempt at encoding dependency information for the structural task of linking quantities with their measured entities, measured properties or qualifiers.

We approach the MeasEval task as a multi-class classification task using a Deep Graph Convolution Neural Network (DGCNN) (Zhang et al., 2018), treating the dependency parse tree as a graph to convolve over. We explore tokenization variants, as well as encodings of the dependency relations using node2vec(Grover and Leskovec, 2016) and UMAP(McInnes et al., 2018) techniques.

2 Problem Statement

The MeasEval (Harper et al., 2021) task consists of 5 (not independent) sub-tasks covering span detection, classification and relation extraction across multiple sentences[2]. Given a paragraph of scientific content in English, a system should: 1) label quantity spans (Q) where Q can be simple count or a numerical value with a unit. 2) if there is a unit it should labelled as Unit(U), and a Q should be classified into one of the types (*count, approximate, range, list, mean, median, medianHasSD, meanHasTolerance, rangeHasTolerance, hasTolerance*) as Modifier(mod). 3) for each Q, systems should identify the span of a measured entity (ME) if one exists and also any measured properties (MP). 4) Identify any spans of qualifiers (QL) that record additional detail related to Q, ME or MP. 5) Label the relationships between Q, ME, MP and QL spans using HasQuantity(HQ), HasProperty(HP) and Qualifies relation types.

3 System Overview

Motivation: Unlike named entity detection tasks, MeasEval's ME or MP detection depends on quan-

[1] https://spacy.io/models

[2] https://competitions.codalab.org/competitions/25770

Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), pages 404–409
Bangkok, Thailand (online), August 5–6, 2021. ©2021 Association for Computational Linguistics

tities and their relation to other tokens within a sentence. Since dependency parse trees are capable of providing approximations of semantic relationships between predicates and their arguments, we opted to generalize over different dependency parse trees to obtain latent path representations to distinguish between different semantic connections that quantities have with MEs or MPs. To encode this path representation we use a Graph Convolutional Networks (GCN) (Kipf and Welling, 2017) in the form of a DGCNN (Zhang et al., 2018), to operate directly on the dependency graph to capture higher order neighborhood information in the form of embeddings. This embedding is used for classifying the relationship type between the tokens to detect one of the 6 classes explained in Section 3.2.2

Our system has 3 main phases: preprocessing, input creation and training a GCN model and post processing respectively. Each phase communicates with the next through CoNLL format (Tjong Kim Sang and De Meulder, 2003) files.

3.1 Phase-I: Preprocessing

We preprocess data using the GATE (Cunningham et al., 2013) modules: ANNIE Tokenizer, ANNIE Sentence splitter, and Stanford Parser (POS tags and dependency graphs (de Marneffe et al., 2006)). Special tokenization rules added are:

mixed character protection prevents splitting tokens of differing character types into different tokens, e.g. *δ13CTOC* $\not\rightarrow$ *δ, 13, CTOC*

split mathematical symbols preserves the usual ANNIE tokenization for $5 \leq 2\theta/^{\circ} \leq 80$ into seven tokens $(5, \leq, 2\theta, /,^{\circ}, \leq, 80)$

number normalization decimal numbers are prevented from being split into 3 tokens. Number words are also identified as numbers

abbreviation period common abbreviations in scientific journals are recognized as integral tokens including the abbreviation period, e.g. *e.g., Fig., sp., spp.* This improves sentence splitting and tokenization

list and interval protection scientific articles frequently report on intervals expressed in different ways and on lists of variable lengths. Both do not usually receive proper parse assignments, because the group as a whole plays a role in the text. To improve the dependency relation assignments, we manually assign the POS tag 'CD' to the groupings:

CD (: | - | *to*) CD
CD (, CD)* *and* CD

unit gazetteer composed from different sources[3] listing 4280 units

3.2 Phase-II: Input creation and GCN training

We train a DGCNN (Zhang et al., 2018) as a mulitilayer neural network that operates directly on a graph to induce node embeddings with properties of their neighborhood. DGCNN takes (A, I) as input, where $A \in \mathbb{R}^{n \times n}$ is an adjacency matrix and n is equal to the number of nodes in the graph G. $I \in \mathbb{R}^{n \times c}$ is an information matrix, associating c feature values with each of the n nodes. A single layer of DGCNN captures information according to:

$$Z = f(\hat{D}^{-1}AIW) \tag{1}$$

where $\hat{D}$ is a diagonal degree matrix with $\hat{D}_{ii} = \sum_j A_{ij}$ (capturing the branching factor of node i) and $W \in \mathbb{R}^{c \times c'}$ is a trainable parameter matrix. f is a nonlinear activation function and $Z \in \mathbb{R}^{n \times c'}$. Higher order neighborhood information is obtained by stacking multiple DGCNN layers:

$$Z^{t+1} = f(\hat{D}^{-1}AZ^{t}W^{t}) \tag{2}$$

where $Z^0 = I, Z^t \in \mathbb{R}^{n \times c_t}$ is the output of the t^{th} graph convolution layer, c_t is the size of the output vector of layer t and $W^t \in \mathbb{R}^{c_t \times c_{t+1}}$.

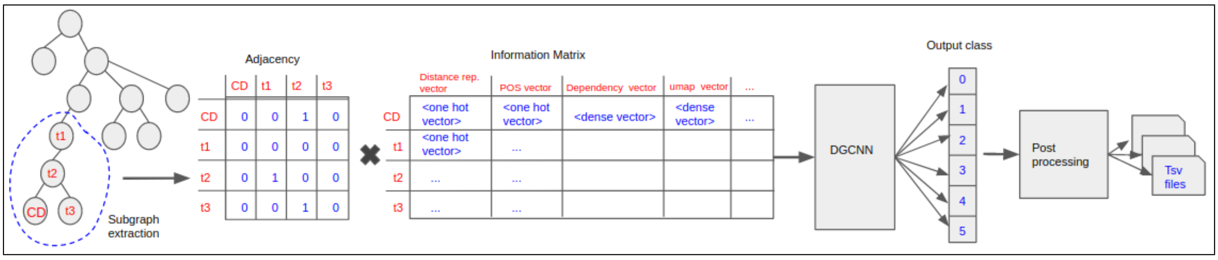
We model a dependency tree as graph $G = (V, E)$, where V are tokens and E are directed dependency relations. We ensure $(v, v) \in E$ for all $v \in V$. To represent paths in the dependency graph between any two nodes, we add explicit reverse links (to *nsubj* from governor to dependant we add *rnsubj* from dependant to governor).

3.2.1 Input creation

The DCNN classifier predicts six output classes, as defined in Section 3.2.2 for token pairs (t_1, t_2), the *subgraph center points*.

The following sections show how (i) candidate token pairs are created, (ii) the smallest subgraph containing t1 and t2 is extracted, (iii) each subgraph SG is represented by (A_{SG}, I_{SG})

[3] http://www.ibiblio.org/units/index.html, https://en.wikipedia.org/wiki/Metric_units



	CD	t1	t2	t3
CD	0	0	1	0
t1	0	0	0	0
t2	0	1	0	0
t3	0	0	1	0

	Distance rep. vector	POS vector	Dependency vector	umap vector	...
CD	<one hot vector>	<one hot vector>	<dense vector>	<dense vector>	...
t1	<one hot vector>	...			
t2	...	...			
t3	...	...			

Figure 1: System Architecture from phase-II to phase-III

Subgraph center point candidates: In all pairs (t_1, t_2), t_1 has to be a CD (a candidate for a quantity).[4]

In principle, all nodes in the graph are candidates for t_2, but we empirically set a limit of five on the connecting path length.

Subgraph extraction For each (t_1, t_2), we select a subgraph containing only the shortest path recursively: SG_1 contains all neighbors of t_1. SG_{k+1} contains SG_k and all neighbors of SG_k. We select the first subgraph that contains t_2.

Adjacency matrix representation The adjacency matrix $A \in \mathbb{R}^{n \times n}$ is a binary matrix. Dependency relations from governor to dependent are one-to-many and all rows in the matrix are interpreted as dependants, the columns as governors.

Information matrix The information matrix I is a $n \times c$ matrix, where c is size of the the concatenated values for the five explicit or latent features associated with each vertex in our system:

DISTANCE FEATURE We use the Double Radius Node Label (DRNL) (Zhang and Chen, 2018) to calculate a combined distance of a node v_i to both subgraph centre nodes within the information matrix as one hot vector as follows:

Nodes t_1 and t_2 carry the label 1. For each v in the subgraph we calculate labels representing distance using the following hashing function:

$$f(v) = 1 + min(d_{t1}, d_{t2}) + \lfloor \frac{d}{2} \rfloor \times (\lfloor \frac{d}{2} \rfloor + [d\%2] - 1) \quad (3)$$

where $f(v)$ assigns labels to all nodes v, d_{t1} and d_{t2} are distances of v with respect to t_1 and t_2 respectively. $d = d_{t1} + d_{t2}$, $\lfloor d/2 \rfloor$ is the integer quotient and $[d\%2]$ is the remainder of d divided by 2.

POS FEATURE encoded as a one hot vector

WORD EMBEDDING from the PubMed ELMo model (Peters et al., 2018) of size 1024.

DEPENDENCY PATH EMBEDDING represents the dependency path of each node within the subgraph from t_1 (base node) $(p(v, t_1))$. We create *dependency embeddings* of size 128 from dependency sequences in the training data using node2vec (Grover and Leskovec, 2016). Given a graph G, node2vec[5] uses a random walk procedure from each node $v \in G$ to produce s sequences of length l, and uses these sequences for training node2vec. We embed dependency sequences instead of node sequences to produce embeddings for each dependency relationship type (i.e. we use node2vec to produce embeddings for edges instead of nodes). To represent $p(v, t_1)$ we concatenate dependency embeddings for each dependecy along the dependency path. Given our empirical limit, $p(v, t_1) \in \mathbb{R}^{5 \times 128}$ and for smaller subgraphs we pad with 0's.

UMAP EMBEDDING As either a complement, or a replacement to dependency embeddings, we experimented with the UMAP dimension reduction technique (McInnes et al., 2018). We trained UMAP as a supervised learning approach feeding dependency embeddings along with class labels and reduced dependency path embeddings to 2 dimensions.

3.2.2 Training the DGCNN model

Input (A, I) was used to train an off the shelf implementation of Deep Graph Convolution Neural Networks (DGCNN)[6] with CrossEntropyLoss[7] as its loss function and with class weights calculated

[4] Note that the gold relations connect text spans, not necessarily tokens. The classifier attempts to predict relations between tokens and the postprocessing phase maps the results to spans.

[5] https://github.com/aditya-grover/node2vec

[6] https://github.com/muhanzhang/pytorch_DGCNN

[7] https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html

by $1/total\_num\_datapoints\_within\_class$. We trained the system for 6 epochs with a batch size of 100 in a cuda environment with 6 output classes to predict.

Class labelling: Center points (t_1, t_2) are predicted to fall into one of six classes:

Class 0: t_1 is not part of a gold Quantity (Q) span[8]; **Class 1:** t_1 is within a Q span but t_2 is not within any gold span; **Class 2:** t_1 and t_2 are in the same Q span (*e.g. 5 kg*); **Class 3:** t_1 is within a Q span and t_2 is any token within the ME span belonging to the same annotation set as t_1; **Class 4:** t_1 is within a Q span and t_2 is any token within the MP span belonging to the same annotation set as t_1; **Class 5:** t_1 is within a Q span and t_2 is any token within the QL span belonging to the same annotation set as t_1.

3.3 Phase-III: Postprocessing

The six classifier classes predict relations between two tokens, while the gold standard annotates relations between text spans. The required mapping to competition output requires several postprocessing steps.

Prediction ranking When multiple t_2 with the same class are predicted for a particular t_1 we choose t_2 with the highest probability.[9]

Span mapping For each prediction $(t_1, t_2) \in$ ClassY we record the token offset for $t2$ as span for ClassY in our system output unless the BERT model from (Therien et al., 2021) finds a neighbouring token with the same predicted class, in which case they are merged into the same span.

Detecting units Once quantity spans are determined, any measurement gazetteer entry within it was labelled as a unit.

Predicting quantity modifiers We used another pretrained BERT model from (Therien et al., 2021) to predict modifiers for each quantity span.

4 Results and Analysis

We split the combined training and trial datasets randomly into 75% training and 25% validation set resulting in 233 and 80 documents in our training and validation set respectively.

We experimented with different preprocessing and feature representations using the official MeasEval evaluation script. DGCNN training parameters were fixed for all the experiments.

Table 1 shows development results, the first row (in italics) shows the competition system. The first column (T) indicates the influence of the list and interval protection step: c indicates it is included, n indicates it is not included. We observe that not including it returns slightly better results, offset by a high rate of duplicates[10].

We experiment with different path length limits (column 4: H). While a length limit of 8 showed better results on the development data than our competition limit of 5, the same is not true for the test data (see Table 2), where there is no equivalent recall gain.

T:	S:		H:	P	R	F1	EM	O
c	s	u	5	*.586*	*.466*	*.520*	*.281*	*.328*
c	s	u	8	.563	.503	.532	.288	.336
n	s	u	5	.567	.485	.522	.291	.334
n	s	u	6	.591	.487	.535	.301	.344
n	s	u	8	**.654**	**.529**	**.585**	**.337**	**.387**

Table 1: Development results. P:precision, R:recall, F1:F1-score, EM:exact match, O:F1 (Overlap)

Table 2 shows competition and post-competition results on test data. The initial competition system is in italics, in bold are the revised results after the organizers removed duplicates.

T:	S:		H:	P	R	F1	EM	O
c	s	u	5	**.546**	**.323**	**.406**	**.217**	**.241**
c	s	u	5	*.554*	*.347*	*.427*	*.232*	*.258*

Table 2: Competition results

Results on the test data are significantly lower, indicating overfitting. Post-competition ablation in Table 3 shows that UMAP, for instance, was not effective. Overlap determined competition rankings.

T:	S:		H:	P	R	F1	EM	O
c	s	u	5	.691	.313	.431	.241	.264
c	s		5	.539	.446	.448	.275	.306
c	s	u	8	.614	.310	.412	.227	.249
c	s		8	.480	.477	.478	.263	.296

Table 3: Post competition results

[8] Not all CDs are part of a gold quantity (e.g. *Fig. 5*).

[9] We exclude predictions for (t_1, t_2) where t_2 carries POS tags among *IN, DT, CD, PUNCT.*

[10] Only in the first row of Table 2 are duplicates removed, all other reported results have duplicates and thus overreport slightly.

	Q	ME	MP	U	Mod	HQ	HP
csu5	**.889**	**.057**	**.007**	**.495**	**.408**	**.028**	**0.**
csu5	*.773*	*.063*	*.007*	*.485*	*.432*	*.028*	*0.*
csu5	.778	.006	0.	.431	.449	.004	0.
cs_5	.830	.177	.177	.449	.484	.167	.053

Table 4: F1 overlap scores of annotation types on competition test data. Q:Quantity, ME:Measured Entity, MP: Measured Property, U:Unit, Mod:Modifier, HQ: Has Quantity, HP: Has Property

Analysing performance for the different labels in Table 4 shows that our system is not yet mature and needs adjusting. The potential of the DCGNN for the tasks is demonstrated by the high results for quantity (Q) and acceptable results for units (U), which are in line with stronger systems. The comparatively low performance for measured entities (ME) and measured properties (MP) demonstrates that the multi-class labelling approach needs better support. We will consider approaches from the literature (Yao et al., 2018), (Sun et al., 2019), (Hong et al., 2020), (Gupta et al., 2016).

HasQuantity and HasProperty received no attention during development and consistently scored 0 for runs with UMAP, see Table 4.

Combined tokens: When *CD(,CD),* and CD* is followed by *respectively*, each list item corresponds to a different entity/property (e.g. *This compares to signatures of accelerated electron precipitation from peaked electrons and "inverted-V" electrons, which occur on 9.8 and 3.4% of MEX orbits, respectively.*) As per gold annotation, 9.8 and 3.4% are 2 different Qs both with "MEX Orbits" as ME and "signatures of accelerated electron precipitation" as MPs. We generate *9.8 and 3.4%* as a single Q and "Signatures" as ME, leading to several false negatives. Documents S0032063312003054-2458, S0016236113008041-3257, S0378112713005288-1916 caused this error.

Math equalities Our system does not protect math environments such as *...tetragonal unit cell with a=4.1816(4) Å and c=10.0322(6) Å...* in document S0022459611006116-1351. Consequently, *a* is labelled as a determiner (DT). As determiners are not part of gold labels, *a* is eliminated in postprocessing when it should have been labelled as an MP in this case and also in document S0022459611006116-1257.

Calculations with measurements Our custom tokenizer does not handle calculations, as in "*...revised average base reaction rate (from* $k1 = 2 \times 10 - 9$ *to* $k1 = 1 \times 10 - 9$ *cm3 s-1)...*" in document S0019103512003533-3908. The gold annotation for Q is $(k1 = 2{\times}10{-}9tok1 = 1{\times}10{-}9cm3s{-}1)$ and ME is *average base reaction rate*, where we return "2×10" as a Q with "average" as ME, "1×10" as Q with "reaction" as ME and "9" (split by symbol "$-$") as seperate quantities with ME as "reaction", incurring false positive errors.

Duplication of Quantities Lists of quantities including units are not combined with our custom tokenizer, thus for *for 4.5 kg and 6 kg samples* in document S0016236113008041-3257, we stipulate two different quantities. This results in duplication of quantities in our submission.[11]

Units The gold standard annotates for instance *thin shale barriers* (S1750583613004192-1126), *das* (S037842901300244X-1654), $\%\Delta E/E$ (S0301010413004096-767), and *KLoC* (S016412121300188X-3207) as units. These are not included in our gazetteer list of 4028 units, incurring false negative errors.

5 Conclusions

The MeasEval task is a challenging task that can benefit from a variety of tools in a well integrated system. The small data size limits a true appreciation of the challenges involved but ablation studies suggest that tokenization variations influence precision and recall differently and should be carefully considered in application systems. Also, a umap reduction suggested a ca 1% performance boost on validation data, but incurs a ca 5% loss on test data, showing signs of overfitting. The subgraph classifier proved effective only for quantity and unit prediction. Ablations show that larger subgraphs and longer paths lead to performance degradation, making a case for more task oriented locality features. Duplicates have to be removed.

While the unusual complexity of the classification task and the limited size of the dataset prohibits very general conclusions, we showed that DCGNNs offer an interesting way to encode dependency information but that it has to be supported by several domain inspired contributions to work for all task components effectively.

[11] This duplication of labels was removed by the organizers for the official ranking.

6 Acknowledgments

We gratefully acknowledge Benjamin Thérien and Parsa Bagherzadeh for their help. The work was supported by NSERC.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python — Analyzing Text with the Natural Language Toolkit*. O'Reilly Media.

H. Cunningham, V. Tablan, A. Roberts, and K. Bontcheva. 2013. Getting more out of biomedical documents with GATE's full lifecycle open source text analytics. *PLoS Comput Biol*, 9(2).

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2537–2547, Osaka, Japan. The COLING 2016 Organizing Committee.

Corey Harper, Jessica Cox, Curt Kohler, Antony Scerri, Ron Daniel Jr., and Paul Groth. 2021. SemEval 2021 task 8: MeasEval – extracting counts and measurements and their related contexts. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*, Bangkok, Thailand (online). Association for Computational Linguistics.

Y. Hong, Y. Liu, S. Yang, K. Zhang, A. Wen, and J. Hu. 2020. Improving graph convolutional networks based on relation-aware attention for end-to-end relation extraction. *IEEE Access*, 8:51315–51323.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *52nd Annual Meeting of the Association for Computational Linguistics*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC 2006*.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. 2018. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Changzhi Sun, Yeyun Gong, Yuanbin Wu, Ming Gong, Daxin Jiang, Man Lan, Shiliang Sun, and Nan Duan. 2019. Joint type inference on entities and relations via graph convolutional networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1361–1370, Florence, Italy. Association for Computational Linguistics.

Benjamin Therien, Parsa Bagherzadeh, and Sabine Bergler. 2021. CLaC-bp at MeasEval 2021. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2018. Graph convolutional networks for text classification. *CoRR*, abs/1809.05679.

Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *arXiv preprint arXiv:1802.09691*.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An end-to-end deep learning architecture for graph classification. In *AAAI*, pages 4438–4445.