CLaC-BP at SemEval-2021 Task 8: SciBERT Plus Rules for MeasEval

Benjamin Thérien, Parsa Bagherzadeh, and Sabine Bergler
CLaC Labs, Concordia University
Montreal, Canada
{b_therie, p_bagher, bergler} @cse.concordia.ca

Abstract

This paper explains the design of a heterogeneous system that ranked eighth in competition in SemEval2021 Task 8. We analyze ablation experiments and demonstrate how the system components, namely tokenizer, unit identifier, modifier classifier, and language model, affect the overall score. We compare our results to similar experiments from the literature and introduce a grouping algorithm developed in the post-evaluation phase that increased our system's overall score, hypothetically elevating our competition rank from eight to six.

1 Introduction

The MeasEval 2021 shared task involves identifying related groups of the four annotation types seen in Table 1, identifying relations linking the annotations of each group, and providing additional information (units and modifiers) for quantities (see (Harper et al., 2021)). Our system learns to classify tokens, groups tokens of the same class into spans and further groups related spans using a distance-based heuristic, providing a baseline for systems that attempt to learn these groupings.

2 Data

The data comprises excerpts from scientific articles of different disciplines: Astronomy, Engineering, Medicine, Agriculture, Biology, Chemistry, Earth Science, Computer Science, and Mathematics. The limited data (see Table 1), the wide range of topics involved, and the idiosyncrasies of each scientific field's vocabulary make the task difficult. In fact, the task organizers report a high degree of inter-annotator disagreement for certain annotation types. Krippendorff's alpha (Hayes and Krippendorff, 2007) for each annotation type is also shown in Table 1, underlining that annotating Measured Properties, Measured Entities, and Qualifiers poses a substantial degree of ambiguity, even to humans.

Annotation	**Frequency**	α
Quantity (QA)	1164	.94
Unit (U)		.86
MeasuredEntity (ME)	1148	.54
MeasuredProperty (MP)	742	.64
Qualifier (QL)	276	.33

Table 1: Frequency of training data annotations and their Krippendorff's Alpha

The training data contains a total of 298 excerpts containing 1164 different quantities. We refer to groupings of annotations as complete data-points (see Figure 1).

All complete data-points contain one quantity and at most one annotation span of each other type. A summary of complete data-point frequencies is provided in Table 2.

Annotations present	**Frequency**
QA	11
ME,QA	361
ME,MP,QA	512
ME,QL,QA	50
ME,MP,QL,QA	225
MP,QA	4
MP,QL,QA	1

Table 2: Frequency of different complete data-points in the training data

3 System

Our system is a pipeline of different machine learning and rule-based modules. We use SpaCy[1] for sentence splitting and tokenization and we fine-tune a SciBERT model[2] to classify each token into one of *Quantity* (QA), *Measured Entity* (ME), *Measured Property* (MP), *Qualifier* (QL), or *Other* (O).

[1] https://spacy.io/

[2] SciBERT uses the same model architecture and pretraining objective as BERT (Devlin et al., 2019), but it is pre-trained on a large corpus of scientific text (Beltagy et al., 2019).

Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), pages 410–415
Bangkok, Thailand (online), August 5–6, 2021. ©2021 Association for Computational Linguistics

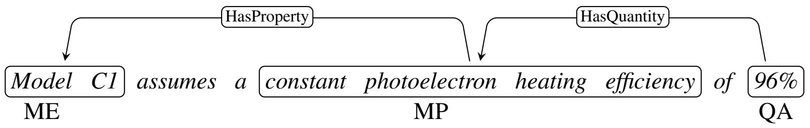

Figure 1: An example of a complete data point with corresponding annotations

During a first postprocessing phase, the token level output of the classifier is merged into spans of adjacent annotations of the same type. We feed the spans annotated as QA to a second SciBERT classifier to determine their modifier class (see Section 3.4). A distance-based heuristic goups the classified spans into complete data-points.

All of the system's deep learning components are implemented using PyTorch (Paszke et al., 2017). Our models use simple (linear + softmax) classification on top of BERT's implementation from the HuggingFace library.[3] Pre-trained weights are obtained from HuggingFace's Model Hub.[4] Our competition SciBERT models are optmized by Adam (Kingma and Ba, 2015) using a constant learning rate of $5e-6$ (the rest of Adam's parameters are set to PyTorch's defaults). The competition system's token classifier is fine-tuned on all the training data for 5 epochs, while the modifier classifier is fine-tuned on all the training data for 3 epochs.

3.1 Preprocessing

Tokenization We modify SpaCy's tokenizer to make several symbols individual tokens. As illustrated in Tables 3 and 4, our tokenizer separates apart mathematical symbols as well as suffixes *Rp, Rs, mH, RRh.*

Prefix	= ∼ ± − ≤ > < ≥
Infix	= ∼ ≈ • : % ( ) → + − ± , > <
Suffix	`Rp Rs mH RRh`

Table 3: Tokenization rules added

The tokenization rules split apart tokens that contain annotations of different types (see Table 4).

Gold span to token conversion The token-level objective used to fine-tune SciBERT requires mapping the gold span annotations onto tokens. However, the gold standard annotations don't always coincide with our token spans. Some gold annotation boundaries are in error (e.g. *oss rate* instead of *loss rate*) and some are deliberate (e.g. *beach* as the annotated part of the token *beaches*).

SpaCy tokens	Modified tokens
∼ 2–3, mA, /, cm2	∼, 2, –, 3, mA, /, cm2
∼ 0.40–0.45, V	∼, 0.40, –, 0.45, V
∼-27%	∼,-, 27, %
(ratio, CaP=1.67)	(,ratio, CaP, =, 1.67,)

Table 4: Tokenization examples

Of the 3330 gold spans that we convert to token-level training samples, only 48 are off by one character, and 2 are off by more than one character. The remaining token level annotations perfectly match the gold spans in character offset. We therefore convert gold annotations to token annotations naively, projecting gold span annotation onto the token span with the least character differences and obtain mappings as illustrated in Figure 2a.

Sentence splitting Scientific text makes frequent use of acronyms (e.g. *fig.*) which confuse SpaCy's default sentence splitter. We use exclusion rules for frequent abbreviations and special tokens that end in punctuation. We also exclude symbols from starting a sentence, as well as tokens not preceded by a period.

3.2 Token Classification

We fine-tune SciBERT to classify tokens into the five output categories QA, QL, MP, ME, and O, where O indicates no annotation for a token. This module processes one document at a time, taking a list of sentences as input, where each sentence is a list of tokens. The output of the classifier is the token annotations for each of the sentences it receives.

Input The special token $[CLS]$ is followed by tokenized sentences separated by $[SEP]$.

Token classification We use a linear layer followed by softmax for the token level multi-class classification into the five label categories from SciBERT output

Out of vocabulary tokens SciBERT's WordPiece tokenizer (Wu et al., 2016) splits unknown

[3] `https://huggingface.co/transformers`

[4] The SciBERT models we refer to are fine-tuned from 'scibert_scivocab_cased' and the BERT model we refer to is fine-tuned from 'bert-base-cased'.

tokens into *subwords* for which it has embeddings. We distribute a token's gold annotation over all subwords within its span (see Figure 2b). To calculate the loss between gold standard and predicted tags, we use cross-entropy. Finally, to predict the class of a token that SciBERT breaks, we take the majority class of its subwords, breaking ties using the class of the first subword, contrary to (Devlin et al., 2019), who always use the class of the first subword.

3.3 Token Span Creation

After token classification, we group any adjacent tokens labelled with the same class into a single span for that label. If the label for a token differs from the label of its neighbors, it forms a span by itself.

3.4 Modifiers

In addition to the five classes discussed so far, MeasEval 2021 also annotates 10 modifier categories[5] for quantity spans: *IsApproximate, IsCount, IsRange, IsList, IsMean, IsMedian, IsMeanHasSD, IsMeanHasTolerance, IsRangeHasTolerance, HasTolerance*, or *NOMOD*.

For instance, the underlined quantity in Example 1 has a modifier class of *HasTolerance* triggered by ±.

(1) *. . . constrain the CTB at 93.90±0.15 Ma*

To determine a quantity's modifier class, we fine-tune a second SciBERT model. Using the $[CLS]$ token as a representation for a quantity's span, the model classifies the span into one of the 11 categories.

Our model predicts at most one modifier per quantity span, which fails in certain examples with more than one modifier. The training data contains 552 quantities with modifiers and 37 of those quantities have two modifiers assigned to them. In competition, our system ranked third for the modifier class.

3.5 Unit identification

We identify units in quantities using a simple rule-based algorithm. No unit is predicted when a quantity span ends in a number or when its predicted modifier is $IsCount$. Otherwise, mark the unit as the string of characters starting from the last occurring numerical character in the quantity span to the end of that quantity span. Example1 highlights the unit in a gray box.

3.6 Span Grouping

We present two approaches for grouping annotation spans into datapoints. The first approach is our original competition algorithm, the second is an improved, post-competition version. Each approach works from lists of token spans (as described above) and outputs a list of groupings. Groupings approximate complete data points.

3.6.1 Original

Our competition system creates candidate groupings for each quantity in a first pass by adding at most one measured entity, one measured property, and one qualifier span to the group if they occur within the same sentence as the group's quantity. A span cannot be assigned to multiple groups. Next, we calculate the character distance between each group's quantity and any still unmatched annotations that are at most one sentence away. The list of distances is sorted and the closest missing annotation within one sentence distance is added. While this algorithm creates some correct complete datapoints, it has two major drawbacks: one measured entity cannot be used for multiple groups and we do not rank the matches to achieve the best fit for all the quantities. When used in our competition system, this algorithm generates 1626 True positives, 892 false positives, and 1504 false negatives.

3.6.2 Span++

In the post evaluation phase, we tested a new algorithm that accounts for token distance during the grouping process. First, the algorithm initializes candidate groups for each quantity. Then, each quantity span is paired with each measured entity span and the shortest token distance between the spans is calculated. The measured entity from the closest pair is added to its quantity's group and the pair is removed from the list. This is repeated for all measured entities that are within a 68 token distance from their quantity. Measured entities farther away are discarded. Then, the same matching process is used for measured properties and qualifiers with cutoff distances of 26 and 25 tokens respectively. Using identical settings to our original submission, we evaluate the performance of our new algorithm (called 'span++' in Table 5). We obtain consistently better overlap f1-scores. While this algorithm supersedes its predecessor, it still

[5] We add an 11th *NOMOD* which indicates the absence of a class.

a)	Tokens:	*Model*	*C1*	*assumes*	*a*	*constant*	*photoelectron*		*heating*	*efficiency*	*of*	*96*	*%*
	Gold tags:	ME	ME	O	O	MP	MP		MP	MP	O	QA	QA
b)	SciBERT tokens:	*Model*	*C1*	*assumes*	*a*	*constant*	*photo*	*##electron*	*heating*	*efficiency*	*of*	*96*	*%*
	Expanded tags:	ME	ME	O	O	MP	MP	MP	MP	MP	O	QA	QA

Figure 2: (a) Converting span annotations of Figure 1 to token annotations. (b) If SciBERT tokenizer breaks a token, we assign its gold tag to all of its sub-tokens

fails to account for multiple quantities having the same measured entity. When used in our competition system, this algorithm generates 1728 True positives, 778 false positives, and 1402 false negatives.

3.7 Relations

Our system uses predefined rules to determine the relations between annotations in a data-point. If there is a measured entity and no measured property, then the measured entity HasQuantity, otherwise, when both are present, the measured entity is assigned HasProperty and the measured property is assigned HasQuantity. Anytime a qualifier is added to a span, we stipulate that it qualifies the quantity.

4 Results

Our submission ranked eighth in competition (listed as rank 7 on the CodaLab leaderboard). Our system is robustly above average on all categories (see Table 5) with strong performance in quantity overlap score (tied for 2nd), qualifier overlap score (2nd), modifier overlap score (3rd), and qualifies overlap score (3rd).

Table 5 shows competition and post-competition results. Our competition system is called *CLaC-BP* and the competition winner is listed under *top ranked*. The first six entries of Table 5 show results for ablation experiments.

Our competition system had a non-zero dropout probability, therefore, we cannot recreate its exact performance for the benchmarking of our ablations. The system labelled **Full** was run with zero dropout probability and is otherwise identical to our competition system. All ablated systems are compared to this baseline.

Noticeable differences The first comparison in row 2 substitutes our fine-tuned SciBERT model for a fine-tuned BERT base model for token classification. SciBERT outperforms BERT for every category except quantity and unit.

In row 3 we assess our modified tokenizer, by substituting it with SpaCy's default tokenizer. The performance is near identical to our baseline system in every category.

The third and fourth systems were trained without modifiers and without units respectively. Overall, the system without units performed worse than the system without modifiers. This is, however, to be expected as there are 905 units and only 552 modifiers, i.e. units account for more overlap score.

Finally, the last comparison showcases the improvements when using *span++*.

5 Discussion

Our system's modular pipeline allows us to assess the components in ablation studies and the various experiments we perform are informative about the task.

Compared to BERT base, SciBERT pretraining gives a solid advantage to all categories, except quantities and units. Not surprisingly, Table 5 shows a greater increase in precision than recall and the exact matches (EM) and F1 overlap (O) scores rise significantly.

Comparing SciBERT's and BERT's performance on the token level objective (Table 6) used during fine-tuning, we see that SciBERT yields greater precision for all competition categories, while BERT yields higher recall for all but *other*. In addition, the token level evaluation is not fully commensurate with the task evaluation, as the significant second task of grouping different spans is not fully assessed.

Our system might benefit from a more precise token level classifier due to its grouping algorithm: it is reasonable to assume that it performs better when more of the detected spans are correct (high precision system) and performs worse with a greater number of incorrect spans (high recall system).

Substituting SpaCy's tokenizer into our system does not make much difference because SciBERT's WordPiece tokenizer will break any unknown to-

System	LM	**EM**	**P**	**R**	**F**	**O**	**QA**	**ME**	**MP**	**QL**	**Unit**	**Mod**	**HQA**	**HP**	**Quals**
Full	SciBERT	.346	.641	.516	.572	.382	.848	.249	.308	.111	.667	.549	.296	.136	.061
Full	BERT$_{base}$	.311	.591	.496	.539	.349	**.853**	.239	.24	.063	**.697**	.537	.253	.107	.038
Full no M.T*	SciBERT	.341	.644	.513	.571	.381	.848	.258	.306	.103	.653	.522	.297	.148	.056
Full no Mod	SciBERT	.309	.63	.464	.534	.347	.848	.249	.308	.111	.667	.0	.296	.136	.061
Full no Unit	SciBERT	.277	.608	.422	.498	.314	.848	.249	.308	.111	.0	**.567**	.296	.136	.061
Full with span++	SciBERT	**.38**	**.687**	**.551**	**.612**	**.421**	.848	**.292**	**.367**	**.151**	.667	.549	**.371**	**.162**	**.081**
CLaC-BP (rank 7)	SciBERT	–	–	–	–	.389	.855	.251	.318	.107	.677	.546	.308	.147	.058
top ranked	–	–	–	–	–	.519	.861	.437	.467	.163	.722	.642	.482	.318	.092
Compet. mean	–	–	–	–	–	.323	.741	.200	.196	.021	.602	.364	.205	.114	.012
Compet. median	–	–	–	–	–	.369	.818	.251	.245	.000	.661	.375	.308	.147	.000

*M.T: Modified tokenizer EM: Exact match HQA: HasQuantity HP: HasProperty Quals: Qualifies

Table 5: Ablation and competition results

kens, however they are tokenized, into subwords at inference time. Yet, because the classifier output is projected onto input tokens, SpaCy's failure to separate certain tokens (see Table 4) might be responsible for the very small degradation when using the SpaCy tokenizer.

Addressing the modifier categories with a separate SciBERT model is successful and allows our system to benefit from several rule-based transition phases. While the task is not well understood (see the overall low performances and the high inter-annotator disagreement), we believe this type of architecture to be more beneficial than end-to-end systems, as we can pinpoint weaknesses and experiment more easily. This is demonstrated by the sizeable improvement of results when implementing a simple distance measure constraint for the grouping step. Once such features have been identified as effective, their encoding in the classifier becomes possible.

	P		R		F1		
Class	S	B	S	B	S	B	support
ME	**.43**	.33	.49	**.55**	**.46**	.41	457
MP	**.47**	.46	.55	**.61**	.51	**.53**	300
QA	**.84**	.81	.96	**.98**	**.90**	.88	950
QL	**.45**	.34	.23	**.32**	.30	**.33**	240
o	**.98**	.98	**.97**	.94	**.97**	.96	12885
M_{avg}	**.63**	.59	.64	**.68**	**.63**	.62	14832

Table 6: Validation SciBERT (S) and BERT base (B) validation set performance after 4 epochs of fine-tuning

6 Summary

Measurements are ubiquitous in scientific articles, yet have so far not been addressed. The MeasEval task is an opportunity to combine different subtasks and experiment how best to combine them. Our system approaches the MeasEval task in a modular fashion: preprocessing, two classifications, and postprocessing. It breaks the task into two, first a classification of the relations for measured entities, measured properties, and qualifiers and second a classification of modifiers. SciBERT, trained on scientific articles, yields better performance than BERT in our experiments, mainly due to improvements in precision. The differences, however are small. Before, between, and after the two classification steps are a number of rule-based modules that create the classifier input and piece together the classifier output for submission. Several experiments on variations in these rule-based accessories suggest their usefulness and ways to improve our results.

References

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*.

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)*.

Andrew F. Hayes and Klaus Krippendorff. 2007. Answering the call for a standard reliability measure for coding data. *Communication Methods and Measures*, 1(1):77–89.

Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceed-*

ings of the 3rd International Conference on Learning Representations, ICLR'15.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS 2017*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.