LIORI at SemEval-2021 Task 8: Ask Transformer for measurements

Adis Davletov
RANEPA, Moscow, Russia
Lomonosov Moscow State University, Moscow, Russia
davletov-aa@ranepa.ru
Denis Gordeev
RANEPA, Moscow, Russia
gordeev-di@ranepa.ru
Nikolay Arefyev
Lomonosov Moscow State University, Moscow, Russia
Samsung Research Center Russia, Moscow, Russia
HSE University, Moscow, Russia
nick.arefyev@gmail.com
Emil Davletov
Katanov Khakas State University, Abakan, Russia
edavletov@yandex.ru

Abstract

This work describes our approach for subtasks of SemEval-2021 Task 8: MeasEval: Counts and Measurements which took the official first place in the competition. To solve all subtasks we use multi-task learning in a question-answering-like manner. We also use learnable scalar weights to weight subtasks' contribution to the final loss in multi-task training. We fine-tune LUKE to extract quantity spans and we fine-tune RoBERTa to extract everything related to found quantities, including quantities themselves.

1 Introduction

SemEval-2021 Task 8 consisted of five subtasks that covered span extraction, classification, and relation extraction tasks. This paper presents solutions to all five of them which showed the best results in the competition[1].

In the subtask 1(A) participants were asked to retrieve *Quantity* (**Q**) spans from texts. For example, in the following text **"The soda can's volume was 355 ml."**, the system should retrieve **"355 ml"** as **Q** span. The rest of the subtasks were to extract information related to retrieved *Quantities* (**Q**s) from subtask A.

The subtask 2(B) was to extract the *Unit of measurement* (**UoM**) of the extracted **Q** and also to classify it into 10 classes: *HasTolerance, IsApproximate, IsCount, IsList, IsMean, IsMeanHasTolerance, IsMeanIsRange, IsMedian, IsRange, IsRangeHasTolerance*. It should be noted that some **Q**s could be related to more than one type and there were ones which didn't belong to any type. The subtask 3(C) was to extract *Measured Entity* (**ME**) and *Measured Property* (**MP**) spans. In the subtask 4(D) additional *Qualifier* (**Qlfr**) spans, which helped to validate or understand the extracted **Q**, were asked to be extracted. And finally, subtask 5(E) was to extract relations between **Q**s, **ME**s, **MP**s and **Qlfr**s.

More detailed information about the competition could be found in the Harper et al. (2021)'s shared task description paper.

2 Related Work

Span extraction and classification problems have a long history of studies and are often studied as a part of Named Entity Recognition (NER). For example, the NER dataset Ontonotes v5 (Weischedel et al., 2013) contains such entities as "Quantity", which also includes measurements, and "Money". However, the general NER approach used in Ontonotes or ConLL 2003 (Sang and De Meulder, 2003) datasets is not so fine-grained as the one that is used in the task under study.

Most state-of-the-art models for named entity recognition and relation extraction are based on Transformer architecture by Vaswani et al. (2017). For example, the top three best models for Ontonotes v5 according to paperswithcode.com use BERT [2]. BERT is a large pre-trained language model based on Attention (Devlin et al., 2019).

[1] https://github.com/davletov-aa/meas-eval

[2] https://paperswithcode.com/task/named-entity-recognition-ner

Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), pages 1249–1254
Bangkok, Thailand (online), August 5–6, 2021.

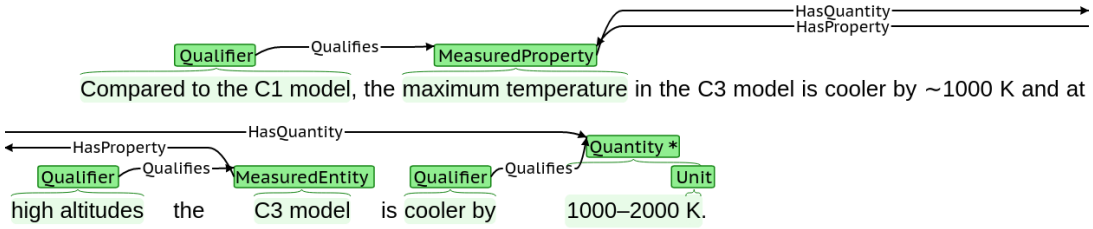

Figure 1: Data example. It shows that one named entity may have several incoming and outcoming relations.

BERT has a unique training procedure where the model is trained using Masked language objective, where some tokens are replaced with a special '[MASK]' token and the model should predict the original token. BERT also had an additional training objective - the model had to predict whether a sentence was random or it followed the first sentence. However, some papers have investigated that BERT is undertrained and that training BERT on more data and for a longer time might increase model performance. RoBERTa was one of the first and more influential papers of such kind (Liu et al., 2019). RoBERTa modifies BERT's pretraining procedure. The RoBERTa model is trained longer, with bigger batches over more data and on longer sequences. RoBERTa's authors have also found that removing the next sentence prediction objective from BERT matches or slightly improves BERT performance. Researchers have also suggested ways of leveraging the nature of the task and adding some problem bias to named entity recognition. Among such works, which is currently the best performing for Ontonotes v5 and CoNLL 2003 according to paperswithcode.com, is LUKE (Yamada et al., 2020). The authors of LUKE have added a new language modeling task that consists of predicting randomly masked words and entities in an entity-annotated corpus retrieved from Wikipedia. The authors have also expanded the self-attention mechanism to entity types and consider entity types when computing attention scores. The proposed approach allowed the authors to achieve state-of-the-art results not only for named entity recognition but for a bunch of other unrelated tasks such as SQuAD1.1 question answering.

For relation extraction, Transformer-based models also outperform other approaches. A promising approach is treating relation extraction as a question answering problem. Among works implementing this approach, we can mention (Cohen et al., 2020) where the authors restructured relation classification as a Question answering (QA) like span prediction problem. It allowed them to get state-of-the-art results for TACRED and SemEval 2010 task 8 datasets.

3 System Description

3.1 Data

The data provided by the organizers contained plain text files and their annotations in tsv format. There were in total 248 training texts, 65 trial ones, and 135 for the evaluation phase. There were 2764, 897, and 1620 annotated entities respectively. The files have been approximately equally distributed among several domains: Agriculture, Astronomy, Biology, Chemistry, Computer Science, Earth Science, Engineering, Materials Science, Mathematics, Medicine. Entities could have been labeled into 5 classes: **Q**, **ME**, **MP**, **UoM** or **Qlfr**.

As input data in the competition was in the form of plain text extracts, we first split them into sentences using PunktSentenceTokenizer and PunktTrainer from NLTK library (Bird et al., 2009). We trained PunktTrainer on texts from the training set. We did data augmentation by including text extracts consisting of two sentences following each other for each text document. So if we had original sentences $[s_1, s_2, s_3, s_4]$ we get an augmented set of texts $[s_1, s_2, s_3, s_4, s_1s_2, s_2s_3, s_3s_4]$. Then we split each example into tokens using RegexpTokenizer from the NLTK library with the following `\w+|\(|\)|\[|\]|[-{.,]|\S+` regular expression. We used the train set for training and the trial set for development.

Also, we relabeled *Qualifier* to *QuantityQualifier* (**QQ**), *MeasuredEntityQualifier* (**MEQ**), and *MeasuredPropertyQualifier* (**MPQ**). By this little trick we solved the problem with examples having multiple **Qlfr**s corresponding to either **Q**, **ME**, or **MP**.

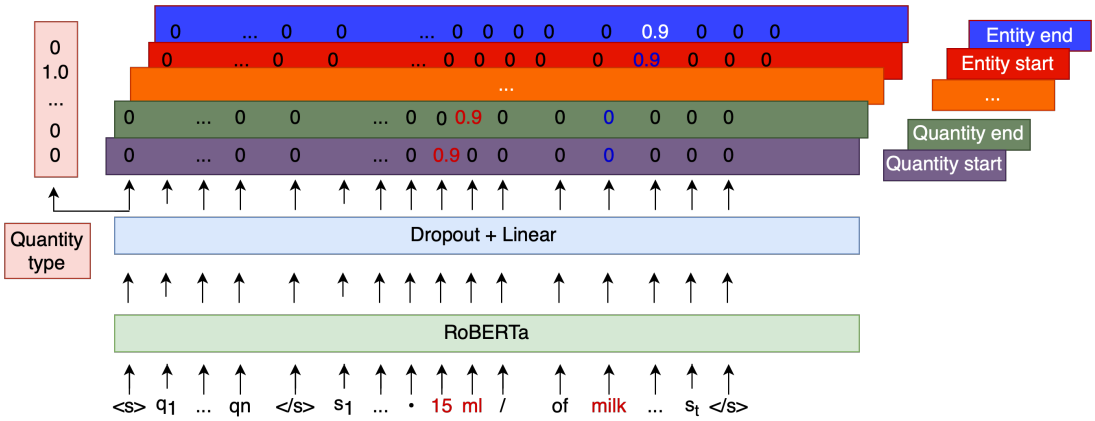

Figure 2: Architecture of the QuAnt system. It takes tokenized text with marked *MeasuredEntity* as input and predicts all needed spans and the class of the *Quantity*.

Hyperparam	Value
dropout	0.1
weight_decay	0.1
warmup_proportion	0.1
lr	1e-4
lr_scheduler	linear_warmup
optimizer	Adam
epochs	50
bs	128
max_seq_len	128
model	xlm-roberta-large
max_grad_norm	1.0
validate_per_epoch	4

Table 1: Training hyperparameters of **QuAnt** system, submitted to competition

3.2 QuAnt System

The architecture of the QuAnt[3] is shown in Figure 2. As could be seen, our model uses the RoBERTa model to extract features for each example. It solves all subtasks of the competition in a multi-task question answering way. We ask our model to predict all BPE subword-level start and end positions of spans (answers) related to **Q** (question). We ask the model by inserting special tokens "•" and "/" around **Q**. Also, the model makes multi-class multi-label predictions regarding the *type of the Quantity* (**QT**).

It takes text extracts containing some **Q**s and positions of the Q regarding which it should make predictions.

For example, for the input text **"The soda can's volume was 355 ml."** and for subword-level positions (6, 7) of the **Q "355 ml"**, the model should predict the following start and end positions: (6, 7) for **Q** (A), (7, 7) for **UoM** (B), (3, 3) for **ME** (C), (4, 4) for **MP** (C), (2, 2) for **MEQ** (D). Also, the model shouldn't predict any label for **QT** (B).

3.2.1 Extract Quantities

So, our approach needs quantity span information as input. And to get that information we went with fine-tuning the LUKE model (Yamada et al., 2020) on the NER task to predict **Q** spans. We used the code provided by the authors of the model. We trained it on the augmented dataset in BIO format with the following hyperparameters: maximum-entity-length, maximum-sequence-length, learning rate, and batch size were set to 64, 256, 1e-5, and 4 respectively. We trained two models with the weight_decay hyperparameter equal to 0.1 and 0.01 for 10 epochs. We were validating our models four times per epoch on the development set and saving the top 3 best checkpoints during training, resulting in a total of 6 models.

So after two training runs, we got 6 trained models. Using the development set we chose the best combination of them for a simple word-level voting ensemble.

3.2.2 Extract Everything

During training and validation, we use **Q** spans from the annotated set. During test prediction, we use spans predicted by the ensemble of quantity extractors from the previous section. Because of our test time augmentation process, we had been able to get up to three entries per each **Q**: for the sentence containing it and for it with either its left

[3]QuAnt - the system deals with quantities in a question-answering-like manner

or right context sentences.

We split tokenized examples into byte-pair-encoding (BPE) subwords with RoBERTaTokenizer which resulted in the following RoBERTa inputs marked by symbols "•" and "/" **Q**s: $[CLS]$ {*Optional question prefix* $[EOS]$} s_1 ... • ***w*** */* ... s_n $[EOS]$.

To vectorize **QT**, we use the output from the last layer corresponding to $[CLS]$ token. And to predict start and end probabilities for each subword of each span type we use outputs from the last layer. We feed them to linear layers to predict **QT**s and span starts and span ends.

During training we optimize the following weighted loss:

$$L_{total} = -\frac{w_{QT}}{bs}\sum_{i=1}^{bs} QT_k^i * log(\hat{QT}_k^i) -$$

$$\sum_{ST \in STs}(-\frac{w_{ST}}{2bs}(\sum_{i=1}^{bs} ST_{start}^i * log(\hat{ST}_{start}^i) +$$

$$+\sum_{i=1}^{bs} ST_{end}^i * log(\hat{ST}_{end}^i)))$$

, where bs – batch size, $[w_{QT}; w_{ST}|ST \in STs] = softmax([w_{qt}; w_{st}|st \in STs])$ – learnable weights vector initialized with ones, QT^i – one-hot encoded **QT** of i-th example (which could be zero vector in some cases and will not contribute during training), $\hat{QT}^i$ – predicted **QT** probability distribution, STs – set of following span types: [**Q**, **ME**, **MP**, **Qlfr**, **UoM**, **QQ**, **MEQ**, **MPQ**], ST_{start}^i – one-hot encoded start position of the corresponding span. $\hat{ST}_{start}^i$ – predicted start positions probability distribution. The same goes for ST_{end}^i.

We trained our model without adding an optional question prefix to RoBERTa inputs. We used hyperparameters from Table 1.

As our test predictions include duplicated predictions for the same **Q** due to the test time augmentation, we remove identical predictions. Worth noting, that there still might be duplicates left in the case of different extracted values for the same **Quantity**. Because of this, our submitted results are higher than the results without test time data augmentation.

So, our model takes **Q** with its context as input and predicts its type and extracts various spans. For all of the subtasks except the subtask E we treated extracted answers as is. For the subtask E we used the following rules to extract relations between **Q**, **MP**, **ME** and **Qlfr** (**QQ, MEQ, MPQ**):

- (**MP**, *HasQuantity*, **Q**);
- if there is **MP** then (**ME**, *HasProperty*, **MP**), otherwise (**ME**, *HasQuantity*, **Q**);
- (**QQ**, *Qualifies*, **Q**), (**MEQ**, *Qualifies*, **ME**), (**MPQ**, *Qualifies*, **MP**);

4 Experiments and Results

In this section, we report the results of our post-evaluation experiments.

First, we experimented with base models. We tried different subtask weighting strategies. As we solve the task in a multi-task way, we need to aggregate the losses of each subtask to optimize the final loss. And here, we tried to just average them (**equal**) or take the weighted sum using learnable weights (**softmax**, **rsqr+log**) vector W with the length equal to the number of training subtasks. In the case of **softmax** weighting strategy we just use softmax over the vector W. In the case of **rsqr+log**, we divide each subtask's loss to its squared learnable weight and sum with the logarithm of it. This approach of weighting subtasks in multi-task learning was introduced by Kendall et al. (2018).

We also experimented with data augmentation. But unlike experiments we did in the evaluation period, here we didn't do test time data augmentation.

Also, we tried to concatenate the question prefix to an input example. We experimented with prefix *Find measured entities and properties of marked quantity*. We hoped it could give extra information to the model regarding the nature of the answer.

Table 2 shows the best results for the development dataset and Table 3 shows corresponding results for the test dataset. Also, there are our official submission results.

Table 2 shows that training time data augmentation improves the overall score. Also, we could see that including prefix question did not improve the overall scores of the models which use data augmentation. Yet we see the opposite picture for the test set in Table 3. It can also be seen that RoBERTa-large not necessarily outperforms RoBERTa base.

We see that using just the average sum of subtask's losses demonstrates the best results.

We also tried to fine-tune the large version of XLM-R with the best hyperparameters from base

Model	Aug	WSch	PQ	O	Q	ME	MP	Qlfr	UoM	M	HQ	HP	Qlfs
post-evaluation phase results: roberta.base													
QA-v1	F	equal	F	45.2	98.9	32.5	36.2	15.2	73.9	66.1	42.4	21.2	9.3
QA-v2	F	equal	T	45.6	98.9	33.2	36.1	15.7	74.3	73.0	42.9	22.6	11.3
QA-v3	F	rsqr+log	F	45.3	98.9	32.3	35.4	15.2	74.7	70.1	41.8	22.0	**12.9**
QA-v4	F	rsqr+log	T	45.8	98.9	33.6	**37.8**	13.4	73.0	67.9	**46.1**	22.5	9.7
QA-v5	F	softmax	F	45.8	98.9	33.4	36.6	13.5	74.0	72.6	42.3	20.5	11.8
QA-v6	F	softmax	T	45.6	98.9	32.5	37.3	16.2	**75.3**	75.5	45.6	21.8	11.3
QA-v7	T	equal	F	**47.7**	98.9	33.1	35.6	16.1	74.3	77.7	40.3	22.7	9.6
QA-v8	T	equal	T	46.1	98.9	32.8	35.5	11.1	73.7	76.4	38.9	21.2	9.3
QA-v9	T	rsqr+log	F	47.6	98.9	32.4	36.8	**18.7**	73.9	**78.3**	40.0	21.9	12.6
QA-v10	T	rsqr+log	T	46.1	98.9	33.1	36.3	14.3	73.9	78.2	42.0	22.3	10.2
QA-v11	T	softmax	F	47.3	98.9	32.9	37.2	16.6	73.6	78.1	41.8	23.0	10.7
QA-v12	T	softmax	T	46.9	98.9	**34.5**	35.2	13.6	73.6	76.3	39.9	**24.2**	10.0
post-evaluation phase results: roberta.large													
QA-v1	T	equal	F	**49.3**	98.6	37.8	38.3	17.7	**75.6**	78.1	43.7	27.0	**10.3**
QA-v2	T	rsqr+log	F	48.8	98.5	35.0	**41.3**	10.7	75.3	77.7	**46.0**	24.5	6.9
QA-v3	T	softmax	F	48.9	98.5	**37.9**	38.1	**17.8**	73.7	**78.4**	44.4	**27.2**	**10.3**

Table 2: Best Overlap F1 scores for the dev set. Aug – augmentation, WSch – weighting Scheme, PQ – Prefix Question, O - Overall, Q – Quantity, ME – Measured Entity, MP – Measured Property, Qlfr – Qualifier, UoM – Unit, M – Modifier, HQ – Has Quantity, HP – Has Property, Qlfs – Qualifies.

Model	Aug	WSch	PQ	O	Q	ME	MP	Qlfr	UoM	M	HQ	HP	Qlfs
evaluation phase results: roberta.large													
QA-v1	T	softmax	F	51.9	86.1	43.7	46.7	16.3	72.2	64.2	48.2	31.8	9.2
evaluation phase results: best results of other competitors													
				47.3	85.5	40.6	43.7	10.7	80.4	61.4	42.4	25.7	6.4
post-evaluation phase results: roberta.base													
QA-v1	F	equal	F	44.8	84.7	38.8	38.1	15.4	66.9	52.0	39.5	24.3	8.5
QA-v2	F	equal	T	43.5	84.7	36.3	34.5	12.3	66.9	57.0	37.6	21.9	5.6
QA-v3	F	rsqr+log	F	45.1	84.7	39.3	39.7	11.9	67.2	50.9	41.7	24.3	7.8
QA-v4	F	rsqr+log	T	45.2	84.7	39.3	40.1	13.8	67.0	48.8	41.7	24.8	7.1
QA-v5	F	softmax	F	43.6	83.8	37.5	35.9	14.9	67.9	49.5	37.9	23.3	8.3
QA-v6	F	softmax	T	42.7	84.7	37.6	32.5	10.2	67.0	55.9	34.4	20.4	5.8
QA-v7	T	equal	F	44.6	84.2	37.4	37.9	9.8	67.4	57.2	39.7	23.6	6.2
QA-v8	T	equal	T	45.7	84.7	39.4	38.8	12.5	66.7	58.8	41.9	24.2	7.4
QA-v9	T	rsqr+log	F	45.9	84.4	39.0	38.3	**18.7**	66.1	58.7	41.5	24.1	**10.6**
QA-v10	T	rsqr+log	T	**47.1**	84.7	39.9	**42.0**	12.5	**68.2**	**59.6**	**44.4**	**26.6**	7.4
QA-v11	T	softmax	F	45.8	84.5	**40.0**	39.9	14.3	66.8	56.1	42.2	24.7	7.7
QA-v12	T	softmax	T	46.0	84.7	39.9	39.3	12.3	67.4	56.2	41.9	26.3	6.6
post-evaluation phase results: roberta.large													
QA-v1	T	equal	F	**48.9**	**84.6**	**43.0**	**45.6**	**15.4**	**67.0**	56.6	**47.7**	**32.0**	8.0
QA-v2	T	rsqr+log	F	47.2	83.8	41.9	42.0	9.2	66.7	58.0	44.6	29.7	6.0
QA-v3	T	softmax	F	47.6	**84.6**	41.9	41.8	**15.4**	66.5	**59.9**	44.9	29.7	**8.3**

Table 3: Overlap F1 scores for the test set. Aug – augmentation, WSch – weighting Scheme, PQ – Prefix Question, O - Overall, Q – Quantity, ME – Measured Entity, MP – Measured Property, Qlfr – Qualifier, UoM – Unit, M – Modifier, HQ – Has Quantity, HP – Has Property, Qlfs – Qualifies.

models. In the case of large models, again, **equal** weighting scheme demonstrated the best result.

In all our post-evaluation experiments we used the same settings as in Table 1. We tried learning rates from $[5e-5, 1e-4, 2e-4]$ and batch sizes from $[32, 64, 128]$.

5 Conclusion

In this paper, we introduced our solution to SemEval-2021 Task 8: MeasEval: Counts and Measurements. Our approach was based on RoBERTa and LUKE models. We show that extracting measurements from a text can be treated as a question-answering task. In this work, we tried a set of different models, hyperparameters, and weighting schemes and present their effect on the final result.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Amir DN Cohen, Shachar Rosenman, and Yoav Gold-

berg. 2020. Relation extraction as two-way span-prediction. *arXiv preprint arXiv:2010.04829*.

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 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*, 23.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. Luke: Deep contextualized entity representations with entity-aware self-attention. In *EMNLP*.