# Report On MeasEval:Counts and Measurements

**Submitted by: Group ID – cs60075_team7**
**Subtask ID – Task 8**
**Github repository:** Somoshree/CS60075-Team-7-Task-8 (github.com)

## Abstract

This document contains the relevant approach, model architectures that we have used, results and further discussions on the task of MeasEval in the SemEval 2021 competition. MeasEval is a new entity and semantic relation extraction task that is focussed on finding counts and measurements, attributes of these quantities and additional information including measured entities, properties and measurement contexts. In this report, we have only discussed on how to approach and further scope of improvement regarding subtasks 1 and 3 of the MeasEval problem statement.

## 1 Credits

The entire credit for making this project a success goes to all the team members of the group cs60075_team7 – **Somoshree Datta (20CS60R05), Pratibha Singh (20CS60R12), Aditya Anand (20CS60R24), Yogesh Porwal (20CS60R52) and Tushika Agrawal (20CS60R55).** The contributions of the various team members are enlisted below.

- **Tushika Agrawal** was responsible for finding the suitable English corpus that needs to be loaded into Spacy for our task of Named Entity Recognition. She also looked over connecting Google Colaboratory to Google drive and finally importing the train and test data from the drive into the Colaboratory environment. Finally, loading all the text data and pre-processing them as required by Spacy NER model was also looked upon by her.
- **Pratibha Singh** was responsible for loading all the tsv data that is needed for training our custom NER model and bringing the data into the correct format as required by Spacy to perform the desired tasks. The training data and evaluation/test data was formatted into a list of tuples of two elements, where the first element is the input text and the second element is a dictionary where key is labelled as "entities" and the corresponding value is again a list of tuples containing "Quantity", "Measured Entity" and "Measured Property" along with their respective spans. She was also involved in constructing all the plots that have been shown in this report at a later stage and also in its analysis.
- **Somoshree Datta** was dedicated towards training the Named Entity Recognizer model of Spacy and making some adjustments as deemed necessary while training on our dataset. She trained the model for 200 epochs, using stochastic gradient descent as the optimizer and a dropout of 0.5 was also kept which is basically a regularization technique to prevent overfitting. Thereafter, the trained model was saved in a pickle file as it might be time-consuming to re-train the model each and every time we try to make predictions on test data. The model that was saved is also reloaded which is then used to make predictions and generate tsv files that are needed for submissions on Codalab.
- **Aditya Anand** devoted his time towards calculating the respective scores and generating the confusion matrix that is needed for the same. He constructed a 4x4 confusion matrix which tells us how well

our model has learned the data on which it was trained. Also, using this matrix, he calculated precision, recall and f1-scores for Quantity, Measured Entities and their respective Measured Properties.

- **Yogesh Porwal** was inclined towards construction of the various plots for precision, recall and f1-score and also analyzed these graphs to get an understanding of how the hyperparameters need to be fine-tuned to make our model better. Also, all the tsv files for the test data has been generated by him which were later submitted in zip format on Codalab. Also, Yogesh has devoted his time towards careful analysis of what might be the future scope of this project and how it could be made much better in the days to come.

## 2 Preparing the work environment

The entire training dataset for this MeasEval task is present in the github repository at MeasEval/data/train at main · harperco/MeasEval (github.com). Since we did this project on Google Colaboratory, so we downloaded the entire training and evaluation dataset into our google drive folder and stored it in the 'Notebooks' folder of google drive in .zip format under the name "measeval.zip", hence facilitating the import from the drive to our Colab environment after unzipping the folder. The train dataset is used to train our model whereas the evaluation dataset is used to test our model. Initially, the text available for each of the documents in the train dataset was converted to a dictionary, where the key of the dictionary was the Document ID and its corresponding value was the text in that particular document. The same was done with the text available for the documents in the eval folder in order to keep the test data ready. All the corresponding labels were also extracted from the given tsv files under the train folder and was stored in another dictionary with Document ID as the key and the corresponding value were the quantities, their spans, measured entities, properties, etc. among other measurements.

## 3 Approach and Model architecture

To solve the specified subtasks, i.e., subtasks 1 and 3 of MeasEval, we have used **Named Entity Recognition (NER)** model for the purpose of information extraction which seeks to locate and classify all the quantities in the text along with their spans, measured entities and measured properties along with their spans too. We have made heavy usage of the **Spacy** library, an open-source software library for advanced natural language processing, in order to accomplish our goal. For our use, we have used large English pipeline pre-trained on written web text (blogs, news, comments), that includes vocabulary, vectors, syntax and semantics. So, we have downloaded the "**en_core_web_lg"** language model for our purpose. Also, Spacy requires the input and labels to be in a particular format. The two dictionaries containing input text and labels were now merged into a single dictionary. We formatted our training data in such a way that it is now a list of tuples of 2 elements, where the first element is the input text and the second element is a dictionary where key is named as "**entities**" (exactly as required by Spacy) and the value corresponding to this key is a list of tuples, each tuple containing "Quantity", "MeasuredEntity" or "MeasuredProperty" along with their respective spans. Also, overlapping entities have been removed from the train dataset as Spacy doesn't work with overlapping entities. We then loaded the large English corpus into Spacy and populated the default NER of Spacy with our own entities that we extracted out from the training dataset that was provided to us. Also, the other pipes of Spacy such as parser, tagger etc. were disabled and only "ner" was kept enabled for solving our problem.

## 4 Training our model

We trained our model for 200 epochs using a dropout of 0.5. We used stochastic gradient descent as our optimizer and trained our model using mini-batches. We are using *minibatch()* function from ***spacy.util*** package to iterate over batches of items one by one. Finally, after each epoch, we use our current model to predict on our test/eval data depending on how much the model has learned till now. For each such prediction on the test data, we calculate the scores and store it in a list which is later used to plot the respective graphs.

## 5 Score calculations

In order to calculate how well our model has learned the data, we determine the confusion matrix and from there we can easily find out the respective precision, recall and f1-score of our model. We use a 4x4 confusion matrix where the 4 rows denote the predicted quantity, predicted measured entity, predicted measured property and predicted none of the three. Similarly, the columns of the confusion matrix denote actual quantity, actual measured entity, actual measured property and actual none of the three. This can be easily visualized from the table shown below:

| Predicted \ Actual | Q | ME | MP | N |
|---|---|---|---|---|
| Q | 0 | 1 | 2 | 3 |
| ME | 4 | 5 | 6 | 7 |
| MP | 8 | 9 | 10 | 11 |
| N | 12 | 13 | 14 | ~~15~~ |

In the above diagram, the entries in the table denote the indices in the one-dimensional list containing 16 values that we have used to represent the 2-dimensional confusion matrix of size 4x4. For example, index 6 in the above confusion matrix denotes the intersection cell of the second row and the third column. Three lists were constructed, one for Quantity, one for Measured Entity and one for Measured Property, each of which contains the tuples corresponding to these 3 entities. If the predicted entities match exactly with the actual entities, then the indices corresponding to 0, 5 and 10 in the confusion matrix gets incremented. Similarly, if the predicted entities don't match with the actual entities, we perform certain actions. For example, if the predicted entity is a Quantity but the actual entity is measured entity or measured property, then we check if the span of the Quantity matches with the span of the actual entity. If it matches, then we increment the corresponding index of 1 or 2 respectively. Else, we increment index 3 of the confusion matrix. This same thing is applied for the remaining two entities too. Finally, we compute the precision, recall and f-score for each of these 3 entities using the given formulas:

**Precision = True Positives / (True Positives+False Positives)**

**Recall = True Positives / (True Positives+False Negatives)**

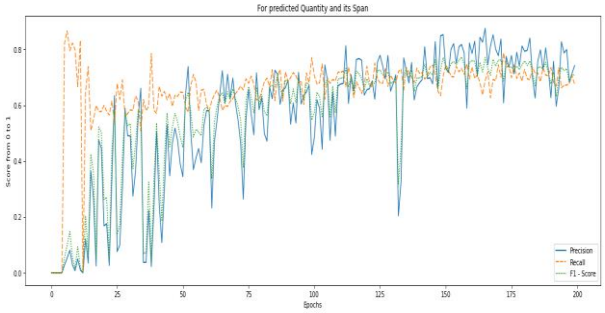
**F1-score = (2*Precision*Recall) / (Precision+Recall)**

## 6 Results

Upon calculating the precision, recall and f-score of Quantities, Measured Entities and Measured Properties on the test data, we get the following plots.

In the three plots immediately below, the dotted blue lines represent the **Precision**, the dotted orange lines represent **Recall** and the dotted green lines represent the **F1-scores.**

**Plot for predicted Quantity and its span**



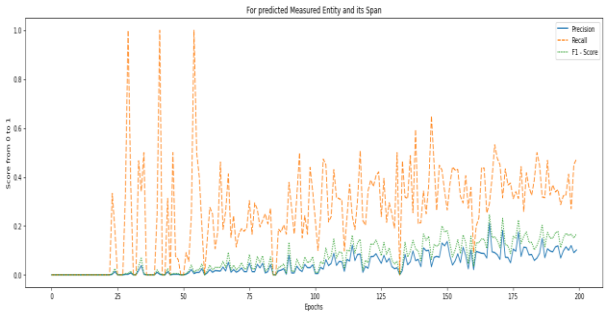

Highest Precision: 0.8759894459102903 at Epoch 166

Highest Recall: 0.8666666666666667 at Epoch 7

Highest F-score: 0.7729468599033816 at Epoch 157

In the above graph, we notice that precision of predicted Quantity outperforms recall.

**Plot for predicted Measured Entity and its span**

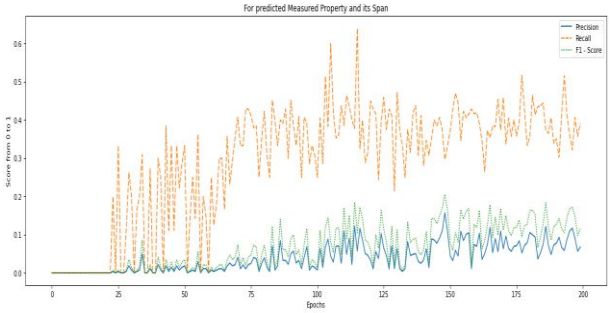



Highest Precision: 0.2111111111111111 at Epoch 167

Highest Recall: 1.0 at Epoch 30

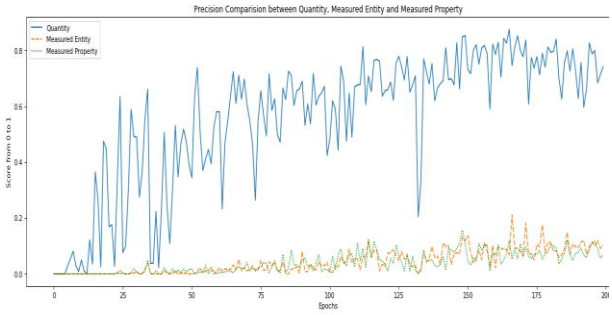
Highest F-score: 0.24516129032258066 at Epoch 167

**Plot for predicted Measured Property and its span**



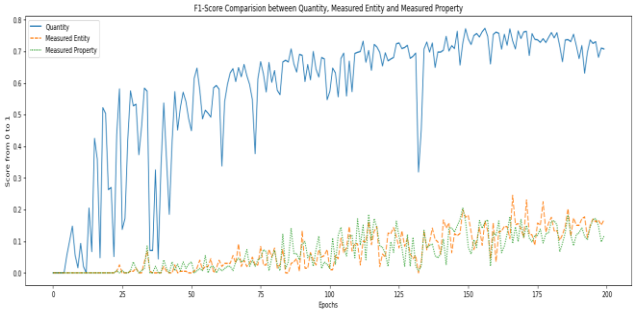

Highest Precision: 0.15695067264573992 at Epoch 149
Highest Recall: 0.64 at Epoch 116
Highest F-score: 0.20527859237536655 at Epoch 149
In the 2 plots immediately above, we find that recall is more than the precision.
In the plots shown below, the blue dotted line denotes the score (precision, recall and f1-score) for Quantity, the orange dotted line denotes the score for Measured Entity and the green dotted line denotes the score for Measured Property.

**Comparing precision curve for predicted Quantity, Measured Entity and Measured Property:**




**Comparing recall curve for predicted Quantity, Measured Entity and Measured Property:**

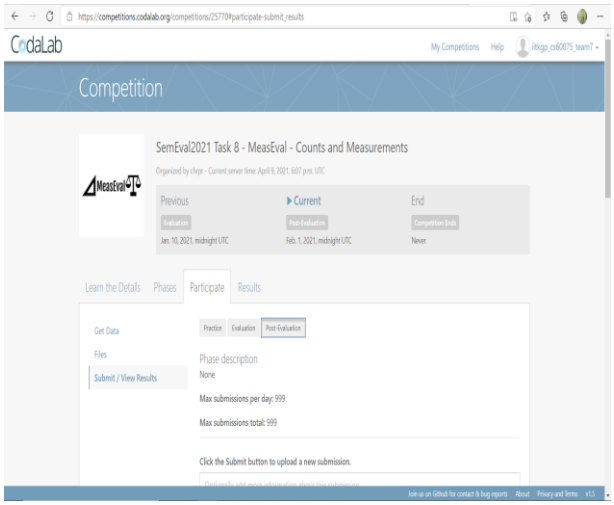

**Comparing F-score curve for predicted Quantity, Measured Entity and Measured Property:**

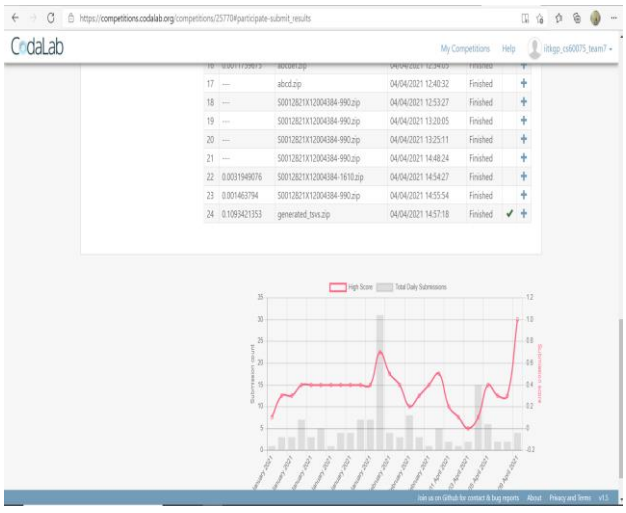



Also, from the above plots, we can see that the f1-score of Quantity is the highest, thereby indicating that our model learned to predict Quantities and its spans quite well. But our model wasn't able to generalize so well for measured entities and measured properties.

## 7 Submissions

We have submitted our project for the Post-evaluation phase in Codalab. The screenshots of the same are attached below:




We submitted in zip format in Codalab submission page for the phase of Post-Evaluation. All the fields

that we haven't predicted, i.e., the subtasks that weren't assigned to us, were filled with dummy values in the tsv files.
The final submission on Codalab (i.e.. entry number 24) gave us a score of 0.1093421353.

## 8 Future Scope

We believe in "There is always a scope of improvement!" philosophy. This is the initial version of the NER system that we have created using the Named Entity Recognizer of Spacy library. But we have already planned many improvements in that.

- Using a BERT (Bidirectional Encoder Representations from Transformers) model for this task which is actually a general-purpose language model that is already pre-trained on a large dataset and need to be fine-tuned using our training dataset for the task of "Named Entity Recognition". It can be used to achieve state-of-the-art performance if fine-tuned well on our custom training dataset.
- Using dependency parsing that provides dependency information between various words in a text encoded in tree form. A dependency parse tree allows one to determine the nature of the relationships between the various components of a sentence. Hence, this can also be used in our application to determine the quantities, their corresponding measured entities and the associated measured properties in a sentence, if it exists. Moreover, dependency parsing can also be used to identify relationships between Quantity, Measured Entity, Measured Property and Qualifier spans using the HasQuantity, HasProperty and Qualifies relation types.

## 9 References


[1] *Custom Named Entity Recognizer* Natural-Language-Processing/Custom_Named_Entity_Recognizer.ipynb at master · srivatsan88/Natural-Language-Processing (github.com)

[2] *Custom Named Entity Recognition with Spacy in Python* Custom Named Entity Recognition with Spacy in Python - YouTube

[3] *Spacy Library for NER model* spaCy · Industrial-strength Natural Language Processing in Python

[4] *Named Entity Recognition with BERT in Spark NLP* Named Entity Recognition (NER) with BERT in Spark NLP | by Veysel Kocaman | Towards Data Science