Medical Visual Question Answering

XX/XX/2019

Avi Turner

[avi.turner111@gmail.com](mailto:avi.turner111@gmail.com)

Table of content

[Abstract 3](#_Toc7792019)

[Preprocessing 3](#_Toc7792020)

[Structuring the data 3](#_Toc7792021)

[Clean and Enrich data 4](#_Toc7792022)

[Feature extraction 4](#_Toc7792023)

[Data augmentation 4](#_Toc7792024)

[Creating meta data 5](#_Toc7792025)

[Model creation 5](#_Toc7792026)

[Creating a model for each question category 5](#_Toc7792027)

[Model(s) training 7](#_Toc7792028)

[Evaluation 8](#_Toc7792029)

[Results submission 8](#_Toc7792030)

[Fine tuning models 8](#_Toc7792031)

[Composing a Model collection 8](#_Toc7792032)

[Evaluating Composition 8](#_Toc7792033)

[Additional attempts 8](#_Toc7792034)

[A single model to rule them all 8](#_Toc7792035)

[Reducing dimensions of answers 8](#_Toc7792036)

[Data Generators 8](#_Toc7792037)

# Abstract

This paper describes the work done in the scope of ImageCLEF 2019 VQA-Med Challenge.

The outline of the work can be described in the following main steps.

1. Preprocessing
   1. Structuring the data
   2. Clean and Enrich data
   3. Feature extraction
   4. Data augmentation
2. Creating meta data
3. Model(s) creation
   1. Creating a model for each question category
4. Model(s) training
5. Evaluation
6. Results submission
   1. Fine tuning models
   2. Composing a Model collection
   3. Evaluating Composition
7. Additional attempts
   1. A single model to rule them all
   2. Reducing dimensions of answers
   3. Data Generators

# Preprocessing

<https://github.com/turner11/VQA-MED/blob/master/VQA-MED/VQA.Python/0_bringing_data_to_expected_format.ipynb>

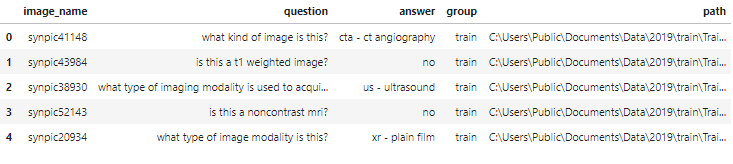
## Structuring the data

The raw input data was composed from a pipe-delimited text with a folder of images corresponding to text lines.



Figure 1: Raw data

The first step was to bring it to a convenient to work with format. We chose to use pandas Data frame.



## Clean and Enrich data

<https://github.com/turner11/VQA-MED/blob/master/VQA-MED/VQA.Python/1_pre_process_data.ipynb>

As part of the preprocessing we removed stop words and tokenized the text.

For Date enrichment, we added a "question category" labeling to data (i.e. question & image pair). This information was given for the train / validation sets and predicted for the test set using thumb rules / NN for the test set.

Another enrichment we have added was a 'diagnosis' label, based on presence of most frequent words in the "Abnormality" Category's answers (Note: Eventually, we did not use this information).

## Feature extraction

Using the NLP library *Spacy*, we have extracted embedding for the questions.

The embedding will eventually be used as the input for the text branch of our model.

For this was have used Spacy's pre trained "en\_core\_web\_lg" vector as described in their home page:

"English multi-task CNN trained on OntoNotes, with GloVe vectors trained on Common Crawl. Assigns word vectors, context-specific token vectors, POS tags, dependency parse and named entities."

In order to reduce data size, the image features are extracted in run time using Keras background Data Generator.

## Data augmentation

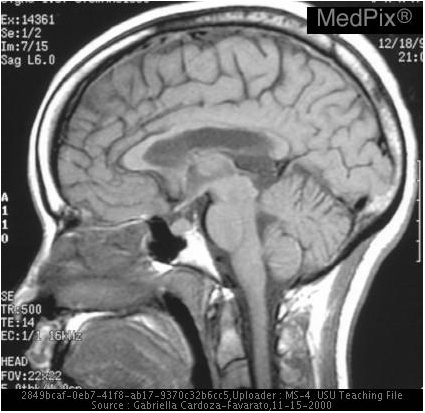
<https://github.com/turner11/VQA-MED/blob/master/VQA-MED/VQA.Python/1.5_data_augmentation.ipynb>

Since the amount of given data seems to be insufficient for meaningful results, we have also used data augmentation in order to get a larger train / validation set.

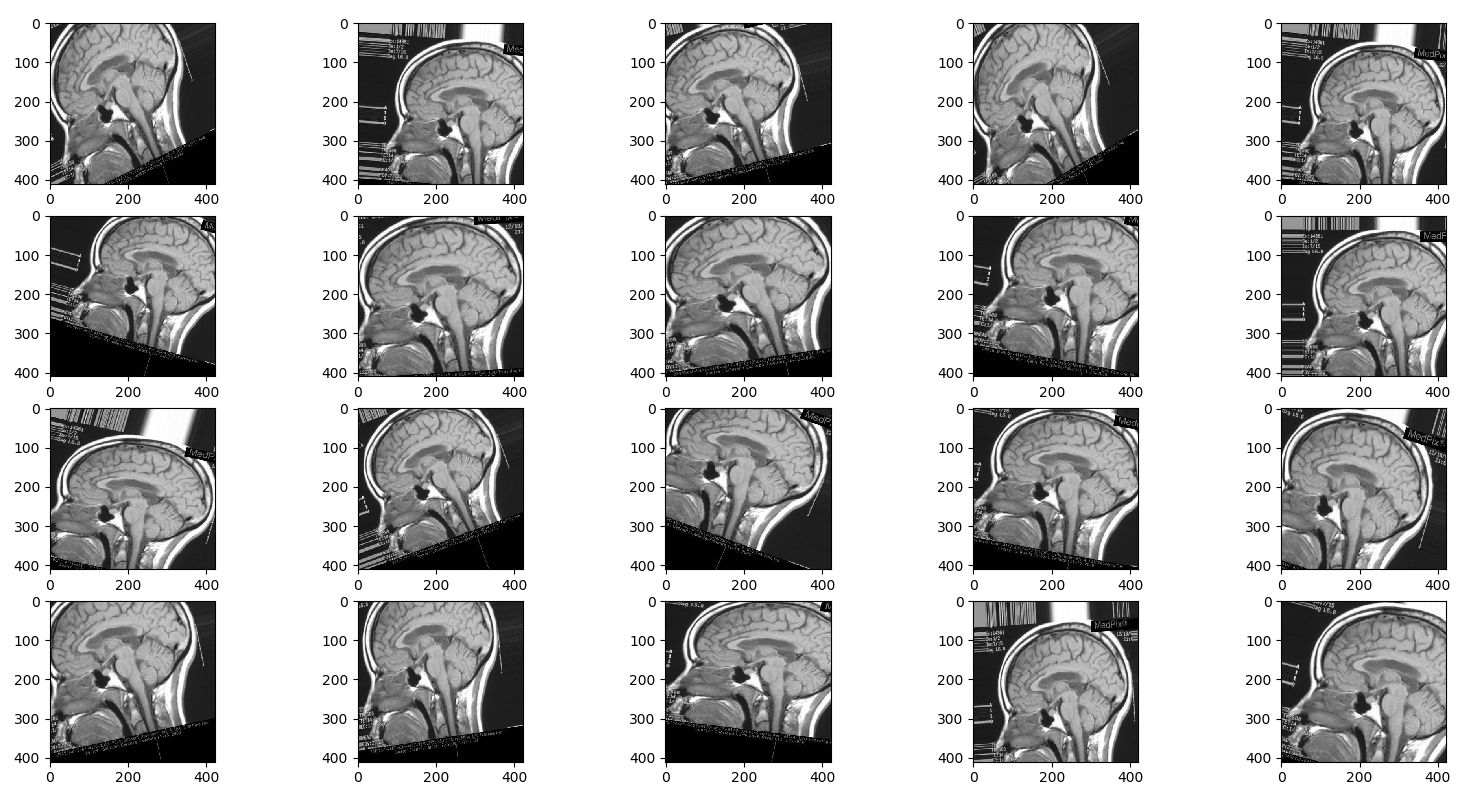
For each question-image pair, we have produced 20 new pairs with the same questioned and a slightly transformed image. The transformation varied in following range:

* rotation\_range: 25 degrees
* width\_shift\_range: 0.15,
* height\_shift\_range: 0.15,
* zoom\_range: 0.15,
* fill\_mode: nearest,

For example:

From: 

We extracted:



# Creating meta data

<https://github.com/turner11/VQA-MED/blob/master/VQA-MED/VQA.Python/2_create_meta_data.ipynb>

The Meta data holds information about which unique words and answers exists in training & validation datasets, and in which categories they appeared (e.g. Modality. Plain etc.).  
Later in the process, this information will allow us to build dedicated models for each category.

# Model creation

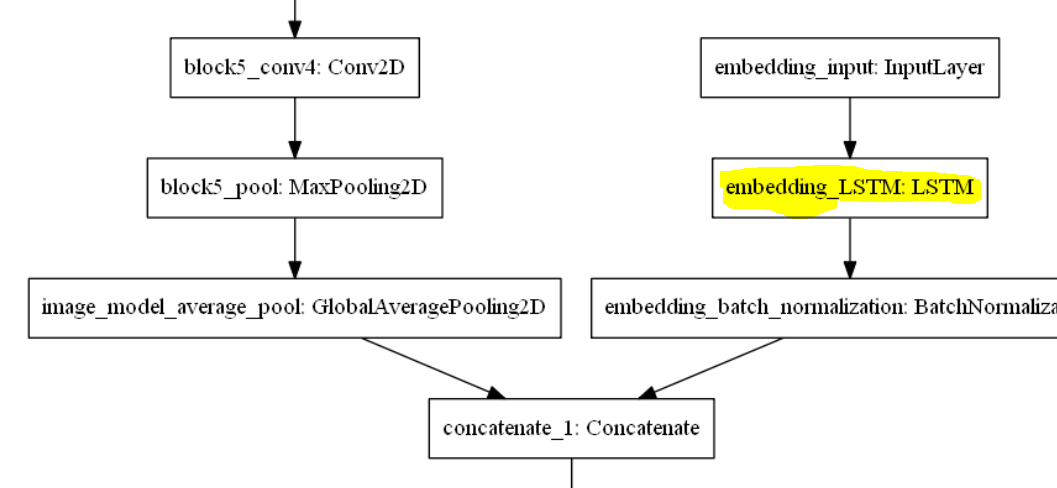
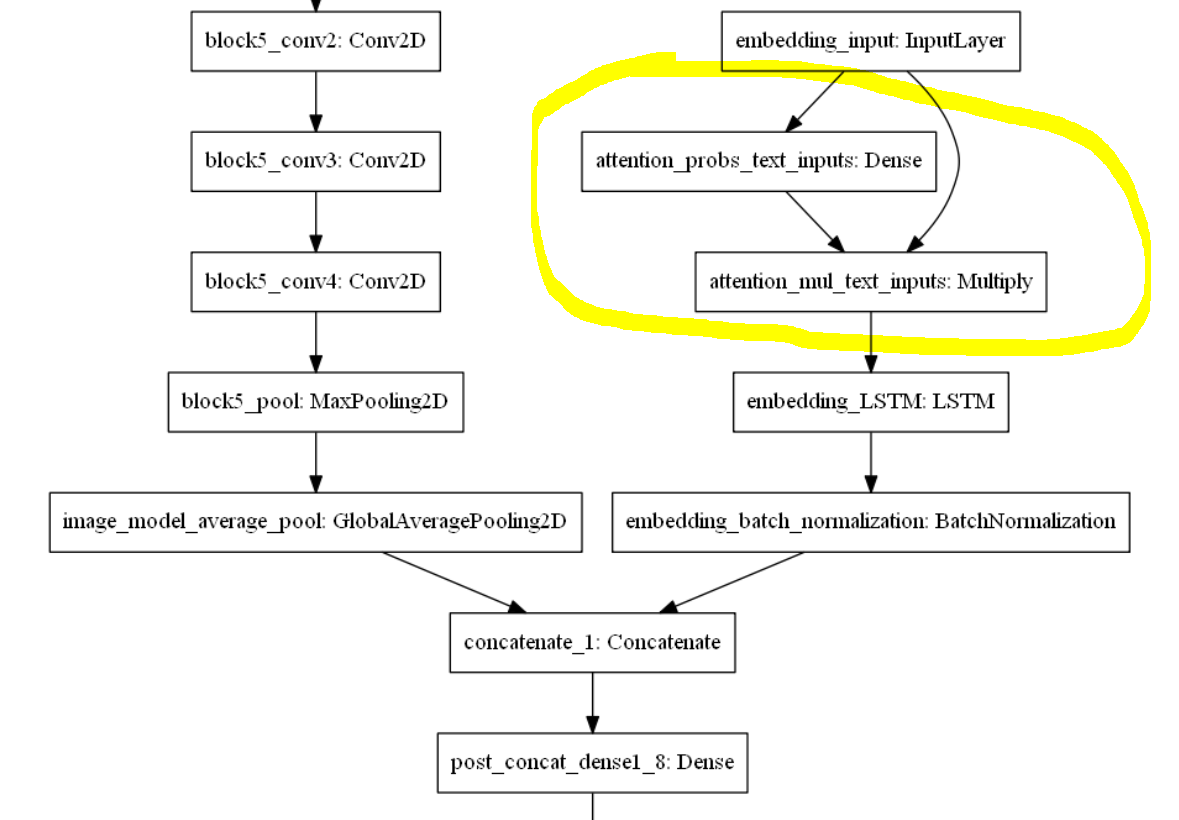
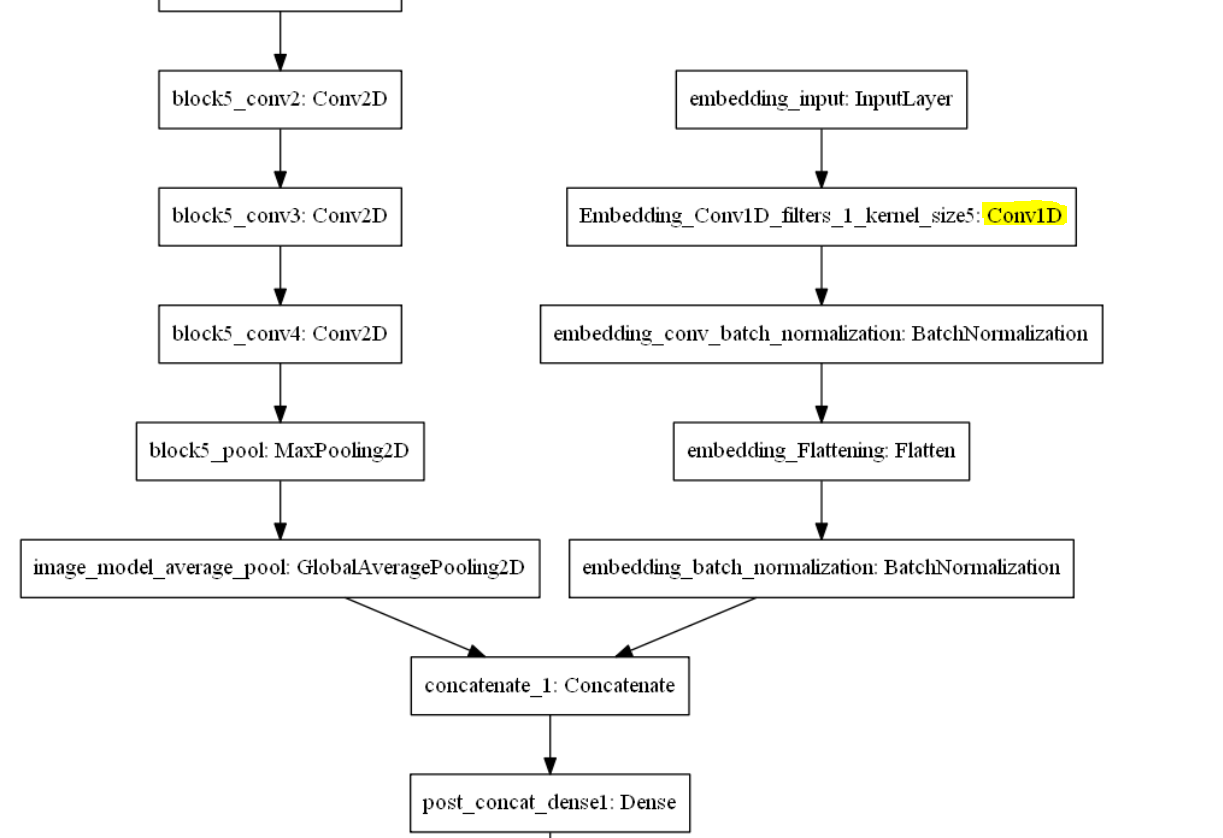
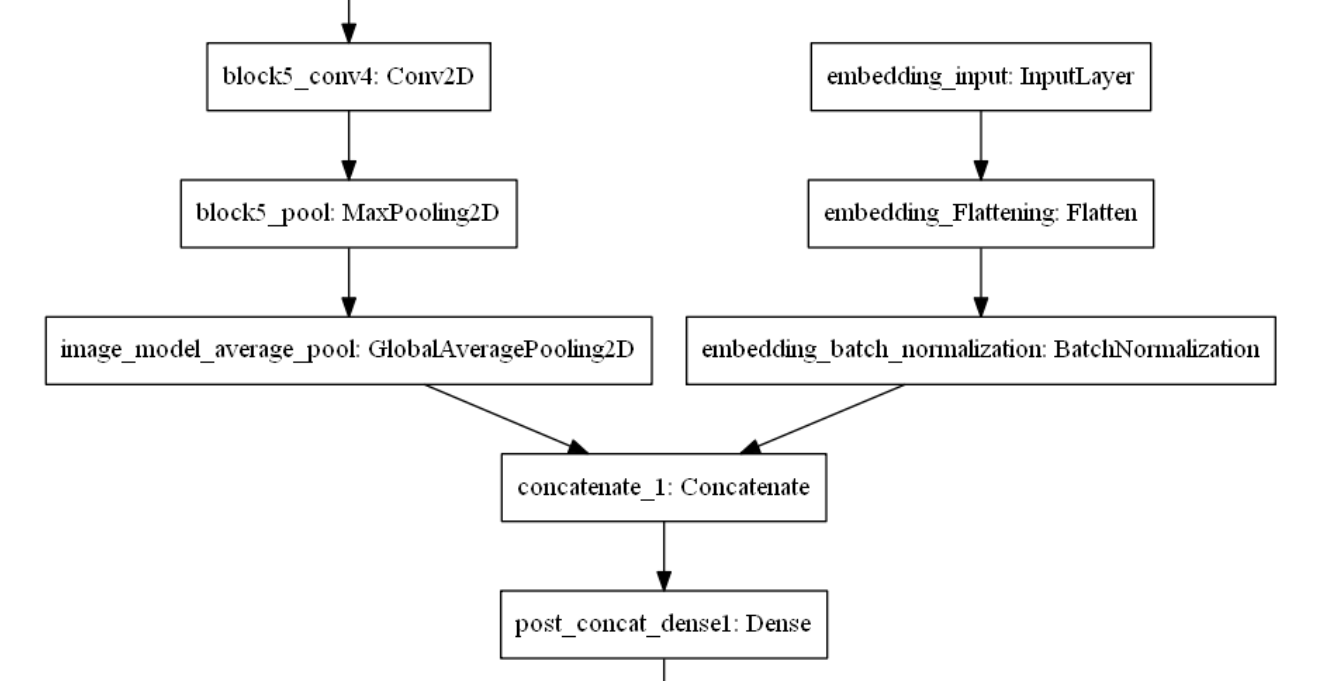
<https://github.com/turner11/VQA-MED/blob/master/VQA-MED/VQA.Python/3_creating_model.ipynb>

When creating the model, we use a basic structure (image branch, text branch, merging the branches, and FC layer(s), and a multi class evaluation output layer).

The parameter that we can play with out of the box are:

1. loss function
2. output activation function
3. LSTM units (will use 'Flatten' instead for 0)
4. Post merge dense units (could be a list to specify multiple layers)
5. Optimizer
6. prediction vector name ('answers' / 'words')
7. question category (for limiting the model to Abnormality / Plane / Organ system / Modality. The default is one model for all).
8. Use text inputs attention (Specifies if attention should be applied for the input of the text branch)

Among other attempts, we tried to use the following techniques in order to achieve better performance:

1. Using LSTM for the text branch 
2. Using Attention layer for the text input (with or without LSTM) 
3. Using convolutional layers instead of LSTM 
4. Just Flattening text embedding without any additional layers 

* It seems that LSTM got more stable models across epochs (Add image)
* Attention / Class weights made things worse (Add Image)
* Did not experiment enough with conv

## Creating a model for each question category

For each of the categories (Abnormality / Plane / Organ system / Modality / Abnormality yes no) we have tried multiple configuration, in order to maximize the overall performance.

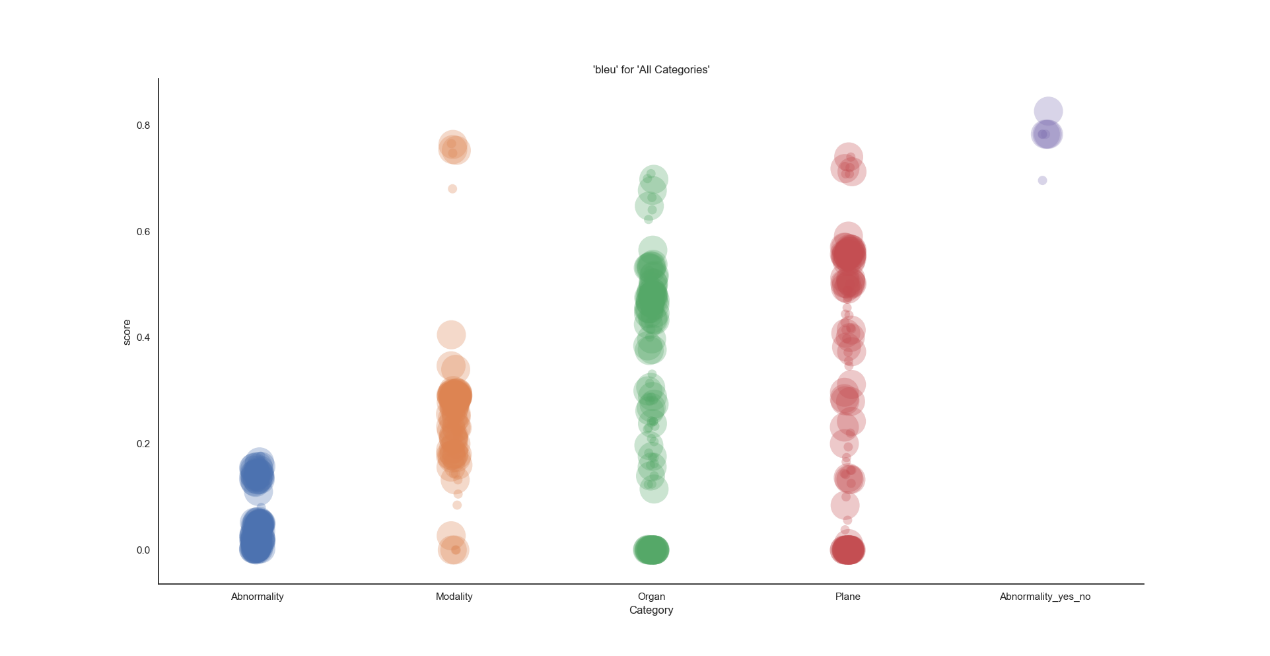


Figure 2: Models Bleu Score by categories. Size is number of dense units

# Model(s) training

<https://github.com/turner11/VQA-MED/blob/master/VQA-MED/VQA.Python/4_training_model.ipynb>

When training the model, we have several parameters we can use for getting better results:

1. augmentations - How many augmentations should be used for each question-image pair.
2. batch size
3. epochs
4. class weight - should class weights be used for compensating for skewed data.

A picture containing text

Description automatically generated

Figure 3: Models Bleu Score by categories. Size is number of dense units

# Evaluation

<https://github.com/turner11/VQA-MED/blob/master/VQA-MED/VQA.Python/6_create_submission.ipynb>

<https://github.com/turner11/VQA-MED/blob/master/VQA-MED/VQA.Python/find_best_history.ipynb>

For evaluating our models, we have implemented 3 evaluators:

1. BLEU (as provided in the 2018 challenge).
2. Strict accuracy (As described in this year's challenge. Since we did not have a 3 human labels, we exact matches got a full score, anything other got none).
3. WBSS (as provided in the 2018 challenge).

At first, we have evaluated each of the models per question categories and per evaluation type.

The first and obvious conclusion we got, was that strict accuracy will work only in models that are trying to predict known answers, and not words. Since it was stated in the google group that this will be the main evaluation criteria, we chose to focus on those models.

Note: For abnormality (unlike other categories) the word prediction gave better results than for the BLEU metric.

Once we got an idea of which models performed better than others, we started tweaking them in small variations to try and improve them (LSTM units / FC layers / epochs).

After it seemed that we got to a glass ceiling for the non-Abnormality categories, we picked the resulting model of the best epoch (Note: We were not worried from over fitting, since the test data was not over fitted).

Once we had models for all of the categories, we have composed from them a "Meta Model" which sliced the data into categories and used a specialist model for every category.

A typical result of a prediction using the Meta model would look as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | bleu | strict\_accuracy | wbss | prediction\_vector |
| Abnormality | 0.055031 | 0.027254 | 0.102724 | answers |
| Organ | 0.708409 | 0.7 | 0.699596 | answers |
| Modality | 0.750807 | 0.712 | 0.702513 | answers |
| Plane | 0.74 | 0.74 | 0.743488 | answers |
| Abnormality\_yes\_no | 0.826087 | 0.826087 | 0.834267 | answers |
| Total | 0.572429 | 0.554 | 0.570493 | -- |

# Additional attempts

## A single model to rule them all

## Reducing dimensions of answers

Before it was announced that struct accuracy will be the main metric for evaluation, we tried to reduce the dimensions of the answers in the abnormality category. The idea was.

Assuming N unique abnormality answers, build an N X N matrix with every row / column index denoted an answer. Then populate the matrix with the BLEU similarity between every 2 answers. For example, the value in cell [j,k] will be the BLUE similarity between answer j and answer k.

Once we had the matrix, we could relate to each row, as the coordinate of the answer in a N dimension space, and then we could use [DBSCAN](https://en.wikipedia.org/wiki/DBSCAN) in order to cluster similar answers.

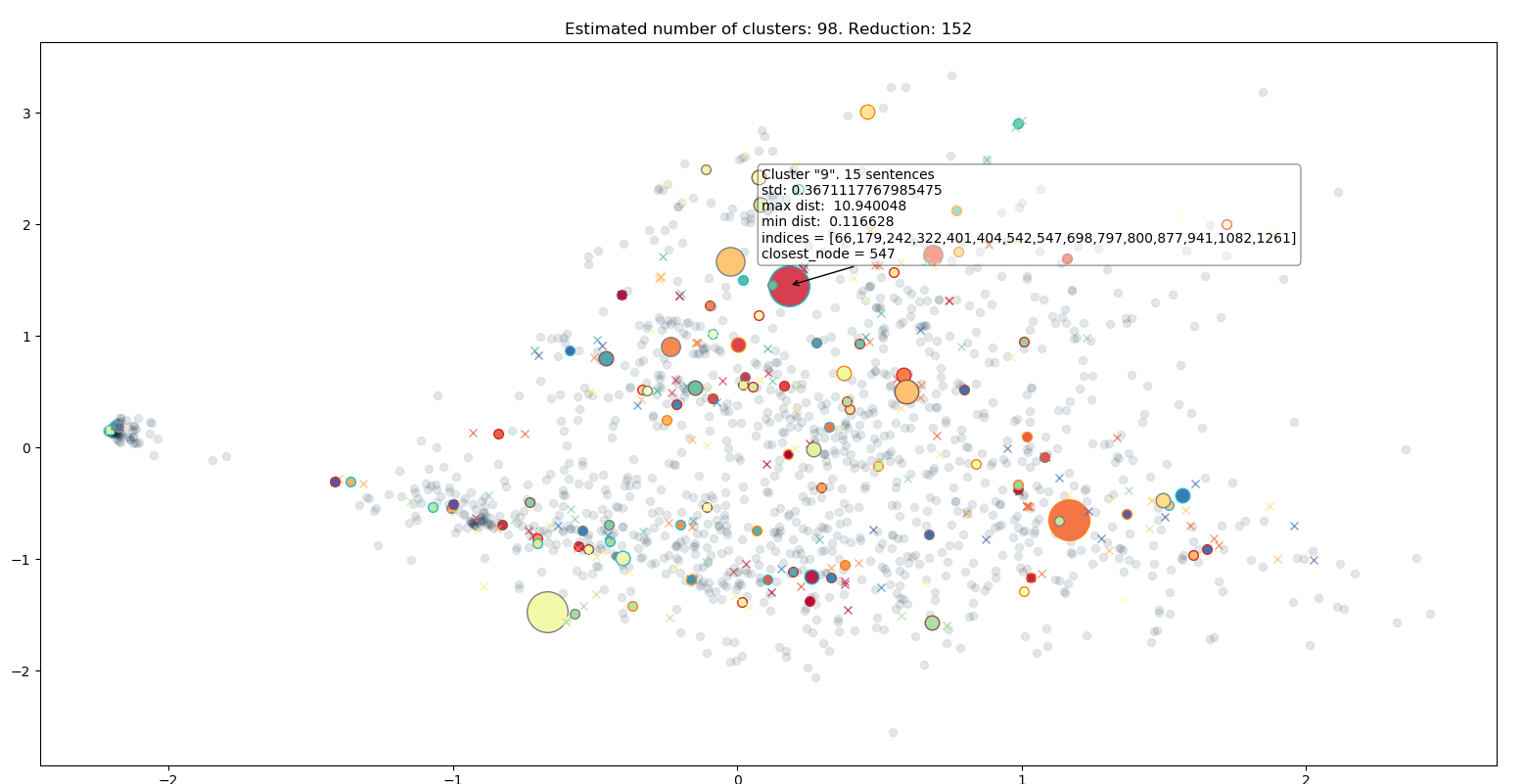


Figure 4:98 Answers cluster. Would reduce rediction space by 152

## Data Generators