Question Generation Project Report.

The Approaches:

The approach used in the project is extremely simple and decently efficient. This is primarily because the alternative powerful approaches required heavy processing time and resources(shy of a couple of days on my mac apparently). However, I'm providing a pseudo description flow of those approaches here. The approaches were defined based on past experience, nlp and squad based research papers and open source projects.

1st Approach - NLP (used in project).**:**
Type: basic nlp and PoS tagging

Description: This approach uses Parts of Speech tagging to define the sentence tree and then generate a question graph based on the nodes and rules.

Improvement: The project uses hardcoded rules.However this approach could have been way more powerful if there existed a dataset with sentence and question pairs(doesn't exist yet but can be synthesised from the squad dataset). This is how that would have worked:

a. Find a large dataset with sentence and question pairs.
b. Tokenize and transform the entire dataset into PoS tags.
c. Convert PoS tags into numeric form using custom vectorization.
d. Use Sequence to Sequence LSTM/GRU encoder-decoder model and tackle the dataset as a supervised learning problem with the sentence vector as input and question vector as target labels.
e. Convert the output back to pos labels.
f. This would generate a model which automatically creates global PoS rules. Which should be improvable to global optimum given enough variance in the dataset.

2nd Approach - Deep Learning (used previously but discontinued due to limited resources):
Type: Sequence2Sequence question generation with gru/lstm encoders

Description: This approach was the one I initially followed and it actually uses the squad dataset. This involved mapping all the words in each record of the squad dataset(100k records) to word embeddings using GloVe(global word vector) or Word2Vec(google's word vectors) which defines connection graphs of each word to it's related words for attention mechanism.

This is useful for getting context of the sentences and allows equations such as woman+king-man = queen, etc.
This failed due to the large size(20gb+ during processing) and processing time(24hours+).
The approach was as follows:

a. Clean the squad dataset into sentence question pairs.
b. Tokenize and vectorize the sentences using glove or word to vec to give them integer values.
c. Split into training and evaluation set.
d. Build a stacked seqence2sequence model using lstm/gru encoder and decoder units.
e. Add the Glove word embeddings layer to the model as an attention mechanism to keep a tab of the context of the sentence.
f. Use the sentence vectors as feature inputs for the encoder and question vectors as target inputs for the decoder.
g. Train the model while tuning hyperparameters- loss/optimization functions and architecture.
h. Save the weights and models to disk and use with novel sentences. Convert output to natural language symbols using glove.