# CS 4740: Intro to NLP
# Project 4: Machine Reading Comprehension

**Xingyu Chen, Tian Ren, Jahanvi Kolte**

**Kaggle Team Name: Gryffindor**

xc374@cornell.edu, tr349@cornell.edu, jsk362@cornell.edu

## I. Baseline

We tried several models to behave as our baseline models. Some of them are mentioned in this section.

**Pre-processing**: We preprocess the data using following rules to make it more consistent and useful for our application

- Have a uniform representation of quotes. Example: ' and `` were replaced by "
- Convert the words to lowercase as case was insignificant for our application [Exception: POS Tagging]
- Remove extra white spaces
- Remove punctuation
- Remove articles (a,an,the)
- Remove miscellaneous unicode characters
- Tokenize the sentences

**Handling unknown and unseen words:** Wherever we observed a word that was not present in the Glove word embedding, we replaced that word with the embedding of '<unk>' key depending on the embedding file and dimensions we used.

**Data Structures:**

We processed the entire training/validation/test dataset and stored the tokenized questions in a list and had corresponding lists to indicate article, paragraph, sentence and if it is answerable list associated with it.

**Model 1: InferSent with Semantic Similarity**

*Description*:

We used pre-trained InferSent to get vector representation and used cosine similarity metric (written by us) with threshold to predict label

*Motivation*:

We wanted to try pre-trained models for sentence embeddings and so we tried InferSent [1] which gives us semantic representation of a sentence. InferSent also uses encoder and is trained on natural language inference data. **Cosine similarity** can be used to evaluate how similar two vectors are and if two sentences have overlapping words with **semantic representation**, then it is likely to answer a question.

*Experimentation*:

We extracted vector representation of sentences (4096 dimensions) from pre-trained Infersent for question and the corresponding sentence in the paragraph that answered the question and computed the cosine similarity of these vector pairs. If it was greater than certain threshold, we marked the label as 1. If any of the sentence in paragraph answers the question, then we mark the label as 1 else the label is marked as 0.

We performed **grid search** on the threshold value and found that 0.74 threshold gave us the best results.

**Idea propagated:**

InferSent was trained using LSTMs on multiple text sources. We believe that training LSTM on given dataset can improve accuracy. So, we will use LSTM to get sentence representation. We can also conclude that semantic information can be useful for us. Perhaps, more complex model will be needed for better accuracy.

**Model 2: InferSent with our own Syntactic Similarity score along with Semantic Similarity score and logistic regression**

*Description*:

We used pre-trained InferSent to get vector representation and used cosine similarity (written by us) as a semantic similarity score. The preprocessed sentences and questions (as strings) are used to compute **syntactic similarity score (**written by us) as discussed in experimentation.

*Motivation:*

We felt that trying to solve the problem only in semantic space is not enough and we need to experiment with common words in sentence and question. We wanted to analyze how **syntactic information** can be useful.

*Experimentation*:

For each **sentence - question pair** for paragraphs, we computed the cosine similarity score on their vector representations from InferSent (4096 dim). We extracted the section question vector pair with **maximum cosine similarity score**. Also, we used nltk to process the **strings** of sentences and questions to **remove stopwords** and replace the words by their **word stem.** For each **sentence - question pair** for paragraphs, we calculated the number of common words and divided that by total

length of question. For each question, we extracted the vector representation of the sentence with **maximum syntactic similarity score**. If any of the sentence was actually the sentence that could answer the question, then the label was marked as 1 or else 0.

Our **final feature vector** included vector representation of question, corresponding sentence with max cos similarity score and max syntactic similarity score, cos similarity score, syntactic similarity score. **12300 dimensions overall.**

**Idea propagated:**
Performance improvement of Model 2 over Model 1 indicates that only semantic information is not sufficient. And so, we should used syntactic information we get from the questions and sentences. So, one of our improved model will try to extract more information from syntax based on POS Tags, TF-IDF, etc.

**Model 3: Bidirectional LSTM Encoders with Neural Net**
*Description*:
Instead of using pre-trained models to generate vector representation of sentences, we used Bidirectional LSTM Encoders and passed the vector representation into neural network for classification task.

*Motivation:*
Both the models discussed till now are feature based. We wanted to try neural models for this task and understand how deep dark knowledge captured by them can help us to analyze if question is answerable or not.

Architecture:
The architecture of system we used for this model is shown in figure

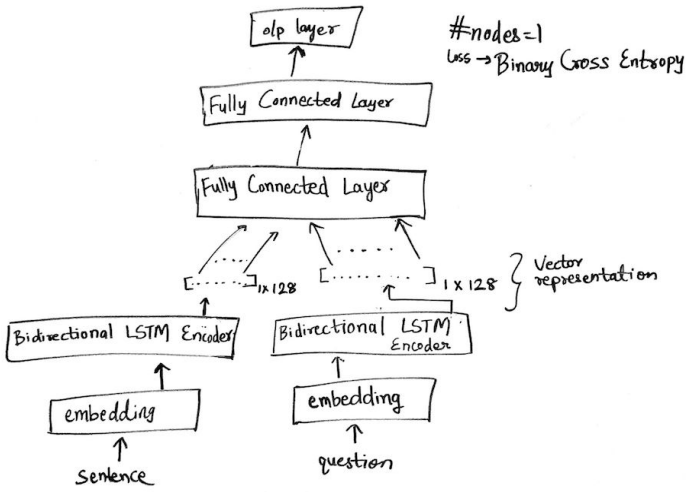



Figure 1: Architecture of Model 3

*Experimentation:*
Using **Glove 300** dimensions word embedding, we generated word embedding for each sentence - question pair in paragraph and passed those individually to two different **bidirectional LSTM encoders**. The output of the last time step of these encoders is passed to a neural network with two fully connected layers to generate the output. (The code has been written by us)

**Idea propagated:**
Using our own LSTM encoders trained on the given dataset, we were able to achieve better performance than other baselines, and it would be worthwhile to try more complex neural models and perform hyper-parameter tuning.
During training, we found that the model does not have significant improvements for the training process between epochs. We assume the problematic point of this model is that we generated biased training data. The encoder takes input from a sentence vector rather than a paragraph vector, so we have to split the training data into sentences and match a specific question. In that way, we are making an assumption that only one sentence in the paragraph contains the answer/paussible answer we are looking for, which is clearly not rigid. In this way, we will end having many '0' labels and few '1', which makes the model harder to train.
**Another model variation**
Also, we tried to concatenate the entire paragraph vector with the question vector, and use a special token <CONCAT> to indicate the boundary between them. Before using an BiLSTM, we used CNN layers to extract features from it. That gave us around 55% percent training accuracy and around 52% validation accuracy. The reason that we didn't proceed with that model is that we thought that the initial simple concatenation like what we did, followed by a CNN lead to huge loss of information.

*** The quantitative/qualitative results and analysis are mentioned in section III, IV and V. It also includes comparison with other models.

## II. Final Model (Improved Models)

**Pre-processing**: We preprocess the data using following rules to make it more consistent and useful for our application

- Have a uniform representation of quotes. Example: ' and `` were replaced by "
- Convert the words to lowercase as case was insignificant for our application [Exception: POS Tagging]
- Remove extra white spaces
- Remove punctuation
- Remove articles (a,an,the)
- Remove miscellaneous unicode characters

- Tokenize the sentences

**Handling unknown and unseen words:** Wherever we observed a word that was not present in the Glove word embedding, we replaced that word with the embedding of '<unk>' key depending on the embedding file and dimensions we used.

**Padding for Model 5:**
To use a standard format, we defined the Bidirectional GRU Encoder to take 600 words for context and 30 words for question. If the size was less than the limit, we padded the embedding for those words with 0. For sizes that were above the size limit; we simply trimmed them. These numbers were finalized based on our analysis of training set for **mini-batch training**.

**Model 4: Logistic Regression and Random Forest on feature based approach using similarity scores, POS and other metrics**
*Description*: We first turned the training data (sentences and questions) into Glove Embeddings with 300D and extract features below based on the sentence-question pairwise embeddings. Then we trained a Logistic Regression model and Random Forest model using this features.
**Final feature vector:**

- TF-IDF features
- POS tag based similarity score
- BLEU-score
- Average word similarity (measure word overlap)
- Average wording embedding similarity
- average paragraph embedding
- TF-IDF weighted paragraph embedding
- average question embedding
- TF-IDF weighted question embedding

The methods of extracting those features are explained in the experiment part.

*Motivation*:
Previous models did not give us a preferred performance, so we decided to switch to feature based approach. Also, the property of question answering suggests that features like POS, NER and words weights would have influence on the task.

*Experimentation*:
We used Glove 300 dimension word embedding to get vector representation of words and generated TF-IDF, POS score and other features using the vector representation. We first tried only with 4 features with LR and RF models. Then we added one feature at a time about different similarities of each sentence-question pair.
For TF-IDF feature, we used sklearn package to extract the popular word (stop word excluded) of the entire corpus. The feature vector is a sparix matrix since we are using a very large corpus
For POS Tag based Score, which we define as a measure of degree of word similarity. We first pos-tag each questions and find out all the words tagged with, 'NN','VBD','VBN','JJ' and set those words as key question words. Then, we went back to the corpus and search the context to find if any of the key words in the question not even occurred in the context. If so, we increment the score by 1.
Those above are syntactical features and we also included **semantic** features like the average of cos-similarity between question and sentences and BLEU score, average of wording embedding for both context and question. We also included cos-similarity of wording embedding and tf-idf weighted wording embedding as semantic features.
For the classider, we have tried logistic regression and random forest. They give similar results but logistic regression is much faster so we used LR in the end.

**Model 5: Using Bidirectional GRUs with Attention**
*Description:*
In this approach, we used bidirectional GRUs as encoders to extract the vector representation of context and question and further added GRUs, additive attention and dense layers for the classification task. As we wanted to use **mini-batch training**, we also constrained the context to 600 words and question to 30 words by using padded sequences.

*Motivation*:
We observed that a simple neural network was giving us around 55% accuracy and so we wanted to try more complex neural models and see if they can capture the deep dark knowledge that might be useful in this classification task.
*Architecture:*
An overview of architecture we used for model 5 is shown in the below mentioned figure

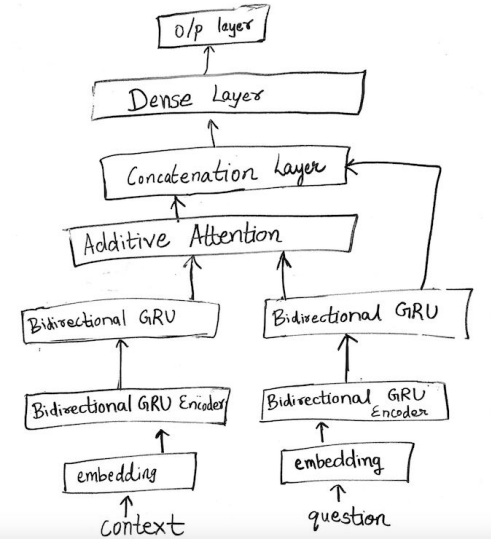


Figure 2: Architecture of Model 5

Hyperparameters of the Neural Model:
Epoch:20
GRU hidden dim: 128
Batch size:128
Embedding dim:300
Callbacks : Learning Rate Reduction, Early Stopping

*Experimentation*:
We used Glove 300d to get vector representation of words in sentence and context and then to generate representation of entire sentence we used separate bidirectional GRUs for context and question.

To use a standard format, we defined the Bidirectional GRU Encoder to take 600 words for context and take 30 words for question. If the size was less than the limit, we padded the embedding for those words with 0. For sizes that were above the size limit; we simply trimmed them. These numbers were finalized based on our analysis of training set for **mini-batch training**.

The output of these GRUs is passed again to another set of GRUs. The output is then concatenated and passed to a dense layer. Our training set had every sentence - question pair in paragraph and corresponding label if question was answerable by that sentence vector using the index. We used Binary Cross Entropy loss in our output layer.

We put 2 seperate parts of GRU layers attached to question vectors and context vectors to extract the feature conveyed in it. Also, we add an additive attention (question to context attention), which is responsible for linking and fusing information from the context and the query words. We used additive attention mechanism and employes, specifically, question to context attention,which signifies which context words have the closest similarity to one of the query words and are thereby critical for answering the query. The context vector from the attention layer is concatenated with the question feature vector from the GRU layer, followed by a fully connected layer with sigmoid activation function. The loss function is binary entropy loss and the inference threshold is set as 0.5.
We experimented the importance of this attention layer. If we deleted the attention and train the model, the validation accuracy is 3% lower than without. The validation accuracy of final model is 63.6%

*** The quantitative/qualitative results and analysis are mentioned in section III, IV and V. It also includes comparison with other models.

*** **The neural models were trained on Google Cloud using GPUs**

## III. Quantitative Results and Analysis

| Model Number | Accuracy | P | R | F1-score | Analysis |
|---|---|---|---|---|---|
| 1 | 0.523 | 0.623 | 0.5131 | 0.5627 | As precision for this model is greater than the recall, we can see that the model is biased towards predicting 1. The model simply relies on cosine similarity and the sentences may have common words but may not necessarily contain the answer of the given question. Moreover, this model uses pre-trained InferSent model which is trained on different natural language but may not be trained on the SQuAD v2 |
| 2 | 0.540 | 0.567 | 0.557 | 0.565 | Accuracy for model 2 is slightly better than Model 1, indicating that we can improve our model using language modeling features. Common words between question and sentence may be one of the indicators for similarity. However, the accuracy is not so good as well which is understandable as having common words is not enough for question to be answerable. |
| 3 | 0.552 | 0.582 | 0.548 | 0.564 | Accuracy of model 3 is better than the previous two models however, it is not proportionate to our expectation from neural models. This can be due to the fact that the data generated is biased for label 0 as for each paragraph, we process question-sentence pair and only assign label as 1 if the question is answerable by that sentence. Thus the model is trained on unbalanced data and is not complex enough to capture enough information to succeed at this task. |
| 4 | 0.612 | 0.605 | 0.614 | 0.609 | This language modeling based improved model uses |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | many features, including TF-IDF, Bleu Score and paragraph embeddings, which contains not only word-wise, sentence-wise and paragraph-wise embedding vector informations, but also considers about syntactic and semantic similarities. Therefore, its performance improved several percentages than the baseline models and did rather well in the prediction. |
| 5 | **0.636** | **0.667** | **0.682** | **0.652** | We get the best accuracy using model 5 which is probably attributed to a complex neural model. This model uses multiple GRUs and Additive Attention and is capable of learning how important the features of encoder output are. Although we trained the model on Google Cloud using GPUs, we still had resource constraints. But we observed that model accuracy improved with epochs. However, we expect a lot of room for improvement if we could design a more complex and better functioned neural architecture and find suitable resources to train it on. |

Table 1: Quantitative Results and Analysis for all models

## IV. Qualitative Results and Analysis

We have mentioned the qualitative analysis of all models in the Table 2. Due to constrained space in report, we do not show the context associated with question.

| **Model Number** | **Input** | **Expected Output** | **Model Output** | **Analysis** |
|---|---|---|---|---|
| 1: Right | What was the ruler at the time of neoclassical introduction? [id: 572996c36aef051400154fff] | 1 | 1 | There are many common words in the question and sentence, so the cosine similarity of the pair are very high, which leads to predicting 1. |
| 1: Wrong | What are ideas shared by the dominite culture? [id: 5ad2e62a604f3c001a3fd924] | 0 | 1 | There are many common words in the question and sentence, so the cosine similarity of the pair are very high, which leads to predicting 1. Since this is the only information the model learns, it does not have other clues for it to predict correctly. |
| 2: Right | What did Margaret Meade use to argue against 19th century racial ideology? [id: 5ad2ac7fd7d075001a429ea4] | 0 | 0 | The syntactic structures of the question and sentence are quite different. The model has a syntactic similarity feature hence it can predict based on the difference of syntax. |
| 2: Wrong | How does Beyoncé say she likes to dress off-stage? [id: 56d4e5022ccc5a1400d832fd] | 1 | 0 | The context sentence is very long for this question and the relevant content is relatively a small part of the long sentence. Therefore, the model gives too much weight when considering the irrelevant but big part. |
| 3: Right | How does Beyoncé say she likes to dress off-stage? [id: 56d4e5022ccc5a1400d832fd] | 1 | 1 | The bidirectional LSTM encoding might have reserved enough information of the sentence context. |
| 3: Wrong | In the judgement, it is stated that the aim of the Genocide Convention, at its most simplest, is preventing the opinions of which victims? [id: 5a6969185ce1a5001a9695fe] | 0 | 1 | There are many common words and similar sentence structures between the question and the context sentence. Hence the neural model might have predicted 1 based on those informations. |
| 4: Right | During what period did the | 1 | 1 | The answer for the question comes from |

| | | | | |
|---|---|---|---|---|
| | earth's crust cooling allow the creation of plates? [id: 57328a3b57eb1f1400fd2d91] | | | several sentences in the context. Since we included features like average paragraph embedding and TF-IDF weighted paragraph embedding, the model is possible to consider discourse between sentences. |
| 4: Wrong | How does Beyoncé say she likes to dress off-stage? [id: 56d4e5022ccc5a1400d832fd] | 1 | 0 | The context sentence is very long for this question and the relevant content is relatively a small part of the long sentence. Therefore, the model gives too much weight when considering the irrelevant but big part. |
| 5: Right | A statute is automatically removed when it is found to be what? [id: 5a79baa417ab25001a89fff7] | 0 | 0 | The reason might be that the neural network with attention has focus in context when finding the result and the focused context may be irrelevant to answer the question. |
| 5: Wrong | During what period did the earth's crust cooling allow the creation of plates? [id: 57328a3b57eb1f1400fd2d91] | 1 | 0 | The answer of the question comes from two sentences and the relevant pieces are of same importance to answering the question. Although our final model has attention mechanism, it still failed to attend 2 places at the same time. Hence it yields the wrong prediction. |

Table 2: Qualitative Results and Analysis for all models

## V. Comparison with baseline and analysis

The quantitative comparison with baseline is mentioned in section III. In this section, we will show some specific examples where we observed improved or deteriorated performance over baseline

We have mentioned the qualitative analysis of baseline in comparison to the improved models in the Table 3. Due to constrained space in report, we do not show the context associated with question.

| **Input** | **Baseline** | **Improved models** | **Analysis** |
|---|---|---|---|
| How does Beyoncé say she likes to dress off-stage? (correct label: 1) [id: 56d4e5022ccc5a1400d832fd] | Model 1 output: 1 | Model 4 output: 0 | Here, we can see that our improved model does worse than the baseline model. This is might because that cosine similarity between sentence and question is very large for "HOW" questions because they tend to have the same structure and words while feature based model might bring in too much noise that affected the result. |
| A statute is automatically removed when it is found to be what? (correct label: 0) [id: 5a79baa417ab25001a89fff7] | Model 1 output: 1 | Model 5 output: 0 | In this example, we can see that our final model did a better job than the baseline model. The reason might be that the neural network with attention has focus in context when finding the result and the focused context may be irrelevant to answer the question. On the contrary, the baseline model got a very high cosine similarity between the sentence and question because it can find a lot of common words. |

Table 3: Comparison of final models with baseline

## VI. Final comments

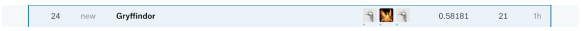
From our experiments, ideally neural model should work better than feature based language models we did with logistic regression. Yet, the final language model has very close prediction accuracy to the neural model we implemented (only 2% difference in terms of accuracy). For the language model, we included lots of feature but also makes it very hard to improve with other additional features. Yet, for neural models, we expect a lot of room for improvement if we could design a more complex and better functioned neural architecture. If we had more time, we might use transfer learning to use a more advanced model:

for example using other pre-trained model with saved layer weights and change last few layers to making it a classification model and only train the parameters on those layers. This approach might give better result and effectively save the training time.

## VII. Kaggle Competition Details

As we observed the best validation accuracy for Model 6, we included that in our Kaggle submission. A snapshot of which is mentioned below.

| **Model** | **Test Accuracy** |
|---|---|
| Model 3 (Baseline) | 52.3% |
| Model 4 (Final Language Model) | 56.3% |
| Model 5 (Final Neural Model) | 58.18% |




## VIII. Team Contribution

We all contributed to each aspect of the project.

| **Project Task** | **Team Members** |
|---|---|
| Pre-processing | Xingyu Chen, Tian Ren, Jahanvi Kolte |
| Baseline | Xingyu Chen, Tian Ren, Jahanvi Kolte |
| Final Models | Xingyu Chen, Tian Ren, Jahanvi Kolte |
| Report | Xingyu Chen, Tian Ren, Jahanvi Kolte |

## IX. References

[1] Supervised Learning of Universal Sentence Representations from Natural Language Inference Data, **arXiv:1705.02364 [cs.CL]**

[2] Question Answering Using Deep Learning, Eylon Stroh, Priyank Mathur

[3] https://github.com/wentaozhu/recurrent-attention-for-QA-SQUAD-based-on-keras