# Homework 4: Sequence to Sequence Modeling (100 points)

Kathleen McKeown, Fall 2019
COMS W4705: Natural Language Processing

Due 12/4/19 at 11:59pm

The aim of this assignment is to provide you with an understanding of how encoder-decoder sequence to sequence models work, give you practical experience training an encoder-decoder model, and show you problems and open areas of research in this domain.

In the first part of the assignment, you will write code and train a encoder-decoder model on a restaurant description dataset. This will be done in a jupyter notebook, and the training *must* be done on a GPU. In the second part, you will do an error analysis and answer questions about open problems and how the model could be improved.

## General instructions

Please post all clarification questions about this homework on the class Piazza under the "hw4" folder. You may post your question privately to the instructors if you wish. If your question includes a partial solution, please post it privately to the **instructors only.**

This is an individual assignment. Although you may discuss the questions with other students, you should not discuss the answers with other students. All code and written answers must be entirely your own.

The submitted written assignment must be typeset, with the except of the model diagram.

**Late policy:** You may use late days on this assignment. However, you MUST include at the top of your submission how many late days you are using. If you don't tell us or have used all your late days, 10% per late day will be deducted from the homework grade. You may choose to save your late days for harder assignments later in the class.

**If you have not already confirmed you can run code on the GPU, budget extra time for this.** Getting set up on VMs, especially with GPUs, is notoriously finicky. Give yourself extra time to work out any problems you may encounter. Additionally, be sure to **shut down your VM instance when not in use**, so you don't run out of credits.

# 1 Coding Portion

See jupyter notebook for solutions.

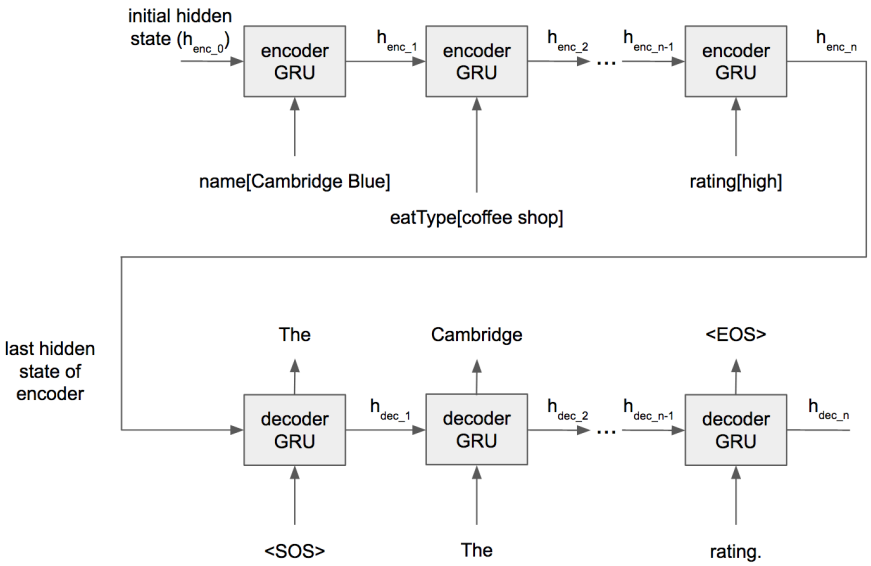
# 2 Written Portion

For each question, write at least 4 sentences.

## 2.1 Model architecture

Draw a diagram of the model used in this homework. It should include at least: individual input tokens, individual output tokens, a start-of-sequence token, an end-of-sequence token, and the final hidden state of the encoder. In words, describe how this model works. (This is the only portion of the written homework that may be hand-drawn, and only the diagram. The description must be typeset. And we would prefer you typeset the diagram if possible.)

**Solution:**




The encoder is a GRU that takes in each token in the input one at a time. The final hidden state of the encoder is used as the first hidden state in the decoder. The decoder always takes in a hidden state and a token and outputs the next token in the sequence (and a hidden state); the first token is an 'artificial' start-of-sequence token and the last token (which indicates to the program to stop generating) is an 'artificial' end-of-sequence token.

### 2.2 Error analysis

Describe three kinds of errors the system makes, with examples (input and output) from the development dataset. You may use greedy decoding for this question. Describe each error in at least 2 sentences.

**Solution:**

There are many kinds of errors:

Repeated words or phrases. The model can get stuck into repeated loops, such as "The Cambridge Blue is is is is" or "The Cambridge Blue is family-friendly is family-friendly is family-friendly". This tends to be helped by beam search, which can get it out of such local optimums.

Missing attributes. The model often ignores certain attributes in the meaning representation, for example not saying what kind of food the restaurant serves. There is nothing about the model that forces it to say everything in the meaning representation.

Incorrect attributes. Perhaps even more often the model gives a slightly off description, like getting the number of stars wrong or changing the average customer rating. Again, there is nothing in the model that enforces correctness.

### 2.3 Beam search analysis

Show your BLEU scores on the development set for greedy decoding and beam search (beam size = 3). For beam search, use the top scoring sequence (i.e. with the highest probability) to calculate the BLEU score for that datapoint. Provide and discuss three examples of beam search. That is, for three input sequences, show the input sequence and multiple outputs from the beam search. How do the multiple outputs differ? Will beam search always include the greedily decoded output? (Why or why not?) What are the advantages and disadvantages of beam search? Discuss the differences in BLEU score for the greedy and beam search decoding.

**Solution:**

BLEU for greedy decoding: 0.4160
BLEU for beam search: 0.4382

```
['name[Alimentum]', 'food[Indian]', 'priceRange[£20-25]', 'customer rating[high]',
'area[city centre]', 'near[Yippee Noodle Bar]']
```

greedy:

<sos> Alimentum is a Indian restaurant in the city centre near Yippee Noodle Noodle Bar. in the Yippee Noodle Bar. Bar. and the customer rating in the city centre near Yippee Noodle Bar. Bar. and the <eos>

beam:
-0.5034 <sos> Alimentum provides Indian food in the £20-25 price range. It has a customer rating of 3 out of 5. It is located in the city centre. <eos>
-0.5023 <sos> Alimentum provides Indian food in the £20-25 price range. It is located in the city centre near the Yippee Noodle Bar. Its customer rating is high. <eos>
-0.4435 <sos> Alimentum provides Indian food in the £20-25 price range. It has a customer rating of 3 out of 5. It is located in the city centre near Yippee Noodle Bar. <eos>

The outputs from beam search are often more fluent than the greedy search, making less grammatical mistakes. However, they also tend to vary in what information is represented. Some miss attributes, and others get attributes incorrect.

Beam search will not necessarily include the greedily decoded output. The greedy output doesn't take into consideration the likelihood of the whole sequence, and so isn't necessarily going to end up in the final beam.

The advantages of beam search are that it can often find better outputs than greedy search, and you can re-rank the outputs of the beam using a secondary scoring function to select the final output using something other than the model's likelihood. The disadvantages are that it takes more computational power, especially for larger beam sizes.

The beam search increased the model score by 2 percentage points, which is a very decent improvement. Likely this would go up more with a custom re-ranking algorithm, like one that explicitly looked for missing or incorrect attributes.

### 2.4 What's wrong with BLEU?

Although several new metrics have been proposed, BLEU remains a common evaluation metric used to evaluate models. What are the problems with BLEU? Does it tend to over- or under-estimate performance? Use examples. What are the benefits of BLEU?

**Solution:**

BLEU has no explicit model of semantics. This means it cannot give high scores to outputs that at semantically correct but don't word-for-word match something in a reference sentence. It tends to favor fluent outputs over correct ones, as a mixture of reference sentences will score highly even if this mixture introduces unexpected meanings.

It tends to over-estimate performance by prioritizing fluency.

However, the advantage is that it's quite easy to calculate and acts as a common baseline between studies. Human evaluations are expensive (in both time and money) and have their own problems, especially around replication. Other automatic evaluations also tend to struggle with capturing correct semantics.

### 2.5 Pyramid scoring for diverse outputs

Pyramid scoring can be used to score generated text outputs, traditionally for summaries. Show how to build the pyramid for the 3 reference sentences shown below. You don't have to build the whole pyramid, but show **two** Summary Content Units (SCUs) that could be added to the pyramid of weight 3, **two** SCUs that could be added to the pyramid of weight 2 and **two** SCUs that could be added to the pyramid of weight 1. Then, assuming that this is the full pyramid, show how you would compute the pyramid score for the generated sentence.

Reference sentences:

1. If you're in the riverside area and are looking for somewhere family-friendly, I've heard that The Cambridge Blue coffee shop has some decent reviews. It's just by Burger King.
2. A low-priced, children-friendly coffee shop, The Cambridge Blue has average customer ratings and is located near Burger King by the riverside.
3. The Cambridge Blue is a family friendly coffee shop providing Italian food with an average customer rating.

Generated sentence:

- Near Burger King at the riverside, there is a family-friendly coffee shop called The Cambridge Blue.

**Solution:**

SCU-3: family-friendly/children-friendly/family friendly
SCU-3: average customer ratings/decent reviews
SCU-2: in the riverside area/by the riverside
SCU-2: near Burger King/by Burger King
SCU-1: low-priced
SCU-1: providing italian food

Let's define $MAX$ as the score of the maximally informative reference sentence. Then:

$MAX_{ref2} = 1 + 3 + 3 + 2 + 2 = 11$
$D = 2 + 2 + 3 = 7$
$D/MAX = 7/11$

### 2.6 How could we deal with unseen restaurant names?

Try generating a sentence with a restaurant name that the model has never seen before. Show the result. How could you change the model to deal with this? Be specific enough that one could try to implement this. (You will not actually implement this, so it could be quite complicated. As long as it's within the realm of possibility, dream big.)

**Solution:**

If you input an unseen restaurant name, the model often selects a restaurant name at random to use instead; normally it's a name that occurred frequently in the training data. (It never gets the name right, as it's not the vocabulary.) One way to deal with this is is to replace all restaurant names with a special token, like RNAME. Then the model will generate RNAME in the correct place in the sentence and you can add the name back in afterwards as a post-processing step. This is called delexicalization.

### 2.7 Bonus: What problems might occur in different languages?

Recently there has been a push to try the E2E challenge in other languages. What problems do you think we might encounter in other languages? Pick at least one language to discuss. Would this challenge be harder or easier in this other language? Why? (You do not need to know another language to answer this question, but you must know enough *about* the language to discuss the impact of its differences.)

**Solution:**

Languages like Czech are more morphologically rich than English. For example, they inflect even proper nouns (the start and endings of the word change based on the syntax of the sentence). This makes the task more difficult, because the model cannot copy as much from the meaning representation. Additionally, solutions like delexicalization or templating no longer solve some of the problems encountered. There is a cool paper called Neural Generation for Czech: Data and Baselines that shows how this task would be done in Czech.