Machine Reading

DSGA 1011 Natural Language Processing with Representation Learning

1 Random QA System

In Question Answering, given a context and a question, we're required to pick the answer for the question from the context paragraph. We'll assume that the answer to the question occurs as a contiguous stream of $k \geq 1$ tokens in the source sentence. Given this assumption, it's enough to pick a starting and ending location to answer a question.

Our random QA system takes in the context paragraph with n tokens and a question. Irrespective of the question, the system picks a starting and ending location uniformly at random from all valid pairs (e, s), where s is the start location and e is end location with e, s satisfying $1 \leq s \leq e \leq n$. For example, $s = e = 10$ would mean that your Random QA system outputs one token ($k = 1$) at location 10 as your answer and $s = 10$ and $e = 12$ would mean that $k = 3$.

1. What is the probability $p_{k,n}$ that our random QA system will output an answer with k tokens when given a context paragraph with n token as input?
2. (Computational) For a fixed n of 100, produce a plot of $p_{k,n}$ and $\hat{p}_{k,n}$ vs k. Design a monte-carlo experiment to estimate the values for $\hat{p}_{k,n}$.
3. Calculate the expected length of the answer given by your random QA system. i,e write an expression for $L_n = E[K]$ (It's fine to leave it as a summation).
4. Plot $\hat{L_n}$ estimated through monte-carlo simulations and L_n for $n = 5, 10, 25, 50, 100, 250$ and 500.
5. Calculate the probability, p_n that Random QA system outputs the correct answer to your question. (Assume that there only one occurence of your answer string in the context paragraph)
6. (Computational) In SQuAD 2.0 data set, the answer for your question can either lie within the context paragraph or there could be no answer within the given paragraph. Let α represent the fraction of questions for which the answer **does not** lie within the paragraph.
   - Estimate $\hat{\alpha}$ from the dataset.

   We can change our Random QA system to take this into account easily. Given a question, with probability α, we say that there is no answer and with probability $1 - \alpha$ we estimate the output of the Random QA system we defined earlier.

   - Implement this Random QA system and run it on SQuAD 2.0 Dev set. Report your mean F1 score and Exact Match (EM) score over 100 runs along with the standard deviation.

2 RNN Based Architecture for SQuAD 2.0

We will use the starter code developed for CS221n. Their project handout is a good source to read if you've questions about the starter code. We highly recommend going to the project handout to get an overview of what we are doing, some details of the model and ideas for Section 2.2.

The provided code implements a baseline architecture. The code is complete and includes the training and evaluation code. Please look at the `readme` of the Github repo on instructions on how to use the starter code.

2.1 Getting Familiar with the Baseline

1. Read `model.py` (and `layers.py`) file and understand the details the model. Summarize the baseline model in the form a diagram.
   Evaluation: We will evaluate the diagram you produce to make sure that you've read and understood the important details of baseline model. During the evaluation, you will be expected to answer any questions on this model.

2. Use the provided `train.py` code and train the baseline model to get the best possible $F1$ on the Dev set. You're free to change any training hyper-parameters (optimizers, learning rate etc). You just have to keep the baseline model fixed.
   Evaluation: We expect you to train your model until you squeeze out what you can from the baseline model (You should be able to achieve an F1 score of about 60 on the provided dev SQUaD 2.0 with about 15 epochs). You're expected to save the model and report the hash code in your accompanying ipynb like you did for the previous homework.

2.2 Let's beat the baseline!

Your task here is to beat the baseline model provided with a new model. The only restriction we have for this question is that you cannot use any model based on ELMo or BERT.

For this section, you're expected to come up with a new model architecture. Note that this is an open ended question and you can implement any idea you have here. Please refer to Sections 5.2, 5.3 and 5.4 of CS221n project handout for some ideas that you can potentially use to improve upon the baseline model.

Evaluation and Deliverable:

1. Provide a diagram of your model like you did for the Baseline model. During the evaluation, we would like you to explain your new model and tell us why you decided to implement such an architecture. Architectural decisions driven by intuition and logic will be evaluated more favorably.

2. **Required Performance:** Your new model should have an $F1$ score of at-least 1 point more than what you got for the baseline model.
   Performance based variable score [10 points]: You will be graded on the curve for 10 points based on your $F1$ score, with more $F1$ score receiving a greater fraction of the 10 points. Note that this part is not bonus. **You should not use your Dev set for training. You will recieve a zero for the homework if you do so.**
   You should save your model and provide the hash code in the accompanying ipynb.

3 Crushing RNN based models: Fine Tuning Bert

In this section, you can use the fine tuning script from Bert Huggingface to use BERT to solve the SQuAD task. We highly recommend reading the original BERT paper to help you solve this part. In this part, you're only required to run the finetuning script from the repository (making sure that you load the data correctly from the starter code in Section 2).

Evaluation: You should be able to show a very high F1 score here. (We expect you to show at least 80). You should hash your saved file and include the code in the ipynb.

4 Analysis

In this section, we will look at some example predictions of the models we built earlier.
Identify and report 3 instances where the baseline model, the model you built in section 2.2 and the BERT based model all give you incorrect prediction. It might be easier to start your search for this example beginning from BERT.