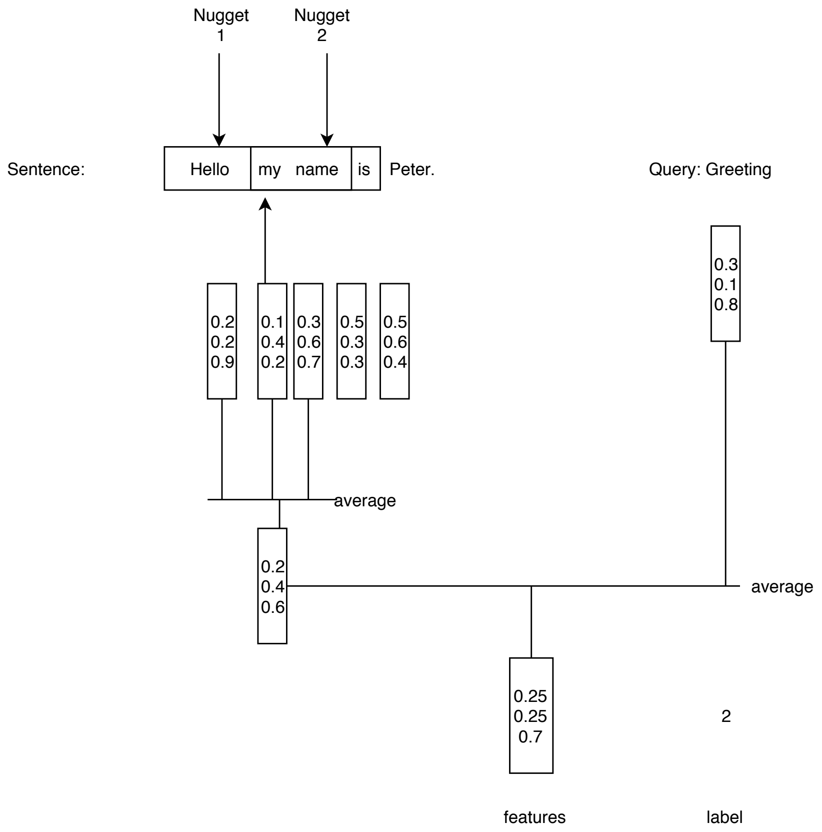
Thoughts about nugget selection

May 11, 2018

1 Single word classification

The first approach I thought about and also already implemented in a simple from splits the task up on a word by word basis. So each word position has to be encoded in a feature representation. I chose a very simplistic form to play around, which builds the average word vector out of the current word and its surrounding neighbours and average that with the average word vector of the query that the topic of the text of which the current sentence is a part of. As labels for each word I have taken the amount of nuggets that this word was part of in this sentence. Those can then be interpreted as class labels. Exam-


Nugget
1
Nugget
2
Sentence:
Hello
my
name
is
Peter.
Query: Greeting
0.3
0.1
0.8
0.2
0.2
0.9
0.1
0.4
0.2
0.3
0.6
0.7
0.5
0.3
0.3
0.5
0.6
0.4
average
0.2
0.4
0.6
average
0.25
0.25
0.7
2
features
label

ple:

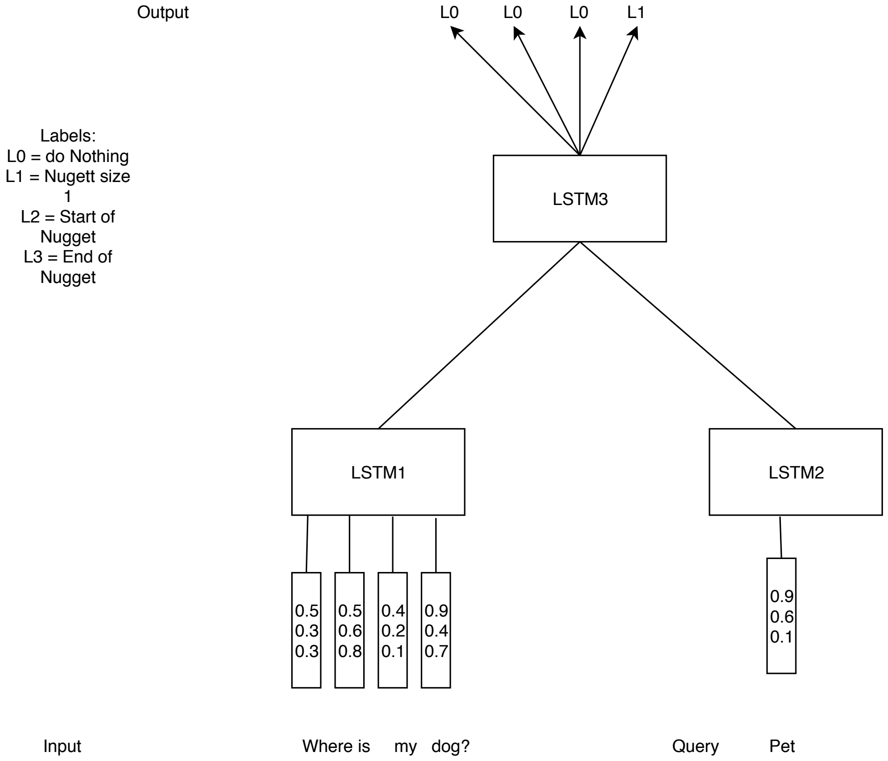
For the prediction phase each word in a sentence of a paragraph would have to be brought into the same feature representation and then gets a assigned a class probability distribution. One could then choose a nugget by combining consecutive words that have a probability that exceeds a certain threshold in the classes that for instance reflect the choice of 3 or more workers.

2 Nugget classification

This approach could work the following. Define a maximum nugget length of a. Then for each sentence in a paragraph form all nuggets that have a length of $<= a$. The labels could either be 0/1 or a multiclass for the amount of workers that actually picked the nugget. The feature representation of the nuggets/query could again be word vectors, or a bag of words representation, bigrams, etc.. For prediction a new sentence would then be split up again in nuggets of length $<= a$ and then brought to the corresponding feature representation. Afterwards there could be a ranking according to the predicted probabilities of the nuggets and then the highest ranking non overlapping nuggets that achieve a certain threshold of probability would be chosen.

3 Sequence to Sequence

This approach would work by using deep learning. First all sentences and the query would have to be transformed to word vectors the same for the topic or query. Afterwards both can be feed into a recurrent layer such as an lstm. The output can then be combined and feed into another recurrent layer that has to predicts a sequence of 3 labels for each word in the sentence where the labels stand for ”do nothing”, ”start of a nugget” and one for the ”end of a nugget”. If we have nuggets of length one we would also need a extra label for that. Visualization:


Output
L0
L0
L0
L1
Labels:
L0 = do Nothing
L1 = Nugett size 1
L2 = Start of Nugget
L3 = End of Nugget
LSTM3
LSTM1
LSTM2
0.5 0.3 0.3
0.5 0.6 0.8
0.4 0.2 0.1
0.9 0.4 0.7
0.9 0.6 0.1
Input
Where is my dog?
Query
Pet