Indian Institute of Technology Delhi

Written Test : Instaminutes

Name: Ritik Kumar Rangari
Entry no: 2019EE30596

Q1. What algorithm or architecture do you think of which can satisfy the requirement (if any)? (Analogy will suffice), bonus point (optional) if you can think of some innovative method outside of present architectures like stacked LSTM, transformers, etc.

Ans - The requirement of the assignment is to find the sentences or phrases that are worth mentioning in the summary, so that a person who wants to quickly review the transcript will get the highlighted summary.

The work of summarizing is challenging for a variety of reasons, some of which are common to other NLP activities, such as translation:

There is no objectively optimal summary for a particular document. Many of them, on average, would be deemed equally good by a human.
It's difficult to pinpoint exactly what makes a good summary and what score we should use to assess it.

Good training data has always been difficult to come by and expensive to acquire.

A summary's human evaluation is subjective, involving judgements such as style, coherence, completeness, and readability. Unfortunately, no score that is both simple to compute and true to human judgement exists at this time. The ROUGE score is the highest we have, but it has significant flaws, as we will show. ROUGE simply counts the number of words, or n-grams, that are shared between a machine-generated summary and a human-written reference summary. It reports a combination of the corresponding recall and precision values.

$$\text{recall} := \frac{\#\text{overlapping } n\text{–grams}}{\#\text{words in the reference summary}},$$

$$\text{precision} := \frac{\#\text{overlapping } n\text{–grams}}{\#\text{words in the machine summary}}.$$

ROUGE-n reports the geometric mean of their combinations (known as the F1 score). Although the ROUGE score does not accurately reflect a human assessment, it has the benefit of computational simplicity and it accounts for some of the flexibility associated with the various summaries that may be produced by rearranging terms within a valid summary.

I have considered the following architecture to be the solution of the given problem statement.

POS Tagging Using RNN

RNN: A recurrent neural network (RNN) is a form of artificial neural network that works with time series or sequential data. Language translation, natural language processing (nlp), speech recognition, and picture captioning are all examples of challenges where deep learning techniques are applied.

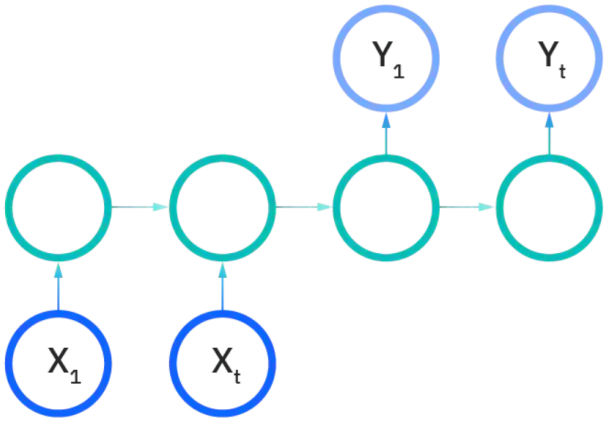
Recurrent networks are further distinguished by the fact that their parameters are shared across all layers of the network. While each node in a feedforward network has a variable weight, each layer of a recurrent neural network has the same weight parameter. To allow reinforcement learning, these weights are still modified through the processes of backpropagation and gradient descent.

Types of recurrent neural networks

Feedforward networks map one input to one output, and while recurrent neural networks are depicted in this fashion in the diagrams above, they do not have this limitation. Instead, the length of their inputs and outputs may vary, and different types of RNNs are employed for diverse applications, such as music production, sentiment categorization, and machine translation.

Different types recurrent neural networks are a)One-to-one b)One-to-many c)many-to-one d) many-to-many

Many-to-many is used for POS.



POS Tagging

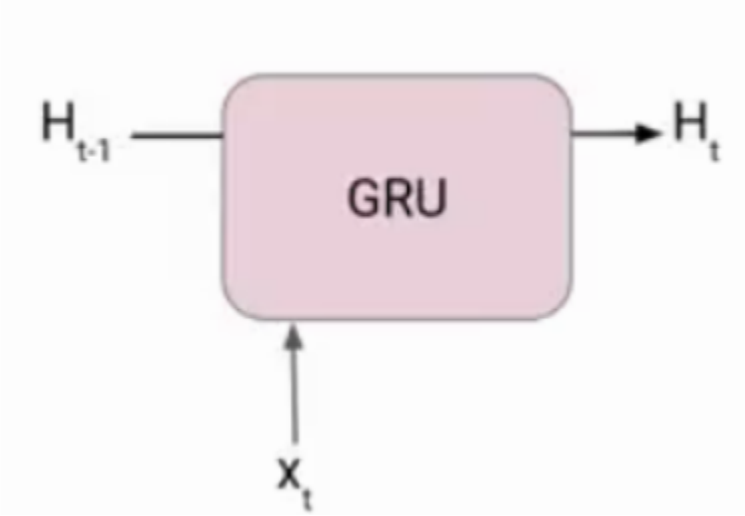
The process of classifying words into their parts of speech and labeling them accordingly is known as part-of-speech tagging, or simply POS-tagging.

Since this is a many-to-many problem, each data point will be a different sentence of the corpora. Each data point will have multiple words in the input sequence. This is what we will refer to as X. Each word will have its corresponding tag in the output sequence. This what we will refer to as Y. Sample dataset.

Q1 part(b)-Optional:

Ans:-

GRUs (gated recurrent units): This RNN version is comparable to LSTMs in that it also aims to solve RNN models' short-term memory problem. It employs hidden states instead of "cell states" to govern information, and instead of three gates, it only has two—a reset gate and an update gate. The reset and update gates, like the gates in LSTMs, determine how much and which information is retained.



It accepts an input Xt and the hidden state Ht-1 from the preceding timestamp t-1 at each timestamp t. It then outputs a new hidden state Ht, which is sent to the next timestamp once more.

In contrast to an LSTM cell, which has three gates, a GRU now has just two gates. The Reset gate is the first, while the Update gate is the second.

The LSTM has three gates, whereas the GRU has just two. The Input gate, Forget gate, and Output gate are the three gates in the LSTM. We have a Reset gate and an Update gate in GRU.

We have two states in LSTM. Long term memory is stored in the cell state, whereas short term memory is stored in the hidden state. There is just one state in GRU, which is the Hidden state (Ht).

Q2. Any automatic (python script) approach can you think of to label the data for training?

Ans. Following are steps required to process the summary.
-> speech recognition
-> speech to text conversion
-> text correction(if any)
-> splitting paragraph into complete sentences with merging the question and its answer in the same token.
-> Applying POS(Part of Speech) recognition to identify different elements of sentences.
->Identifying Important words or chunks from the sentence which can be considered as Insights and if yes, then highlight it.
By the above steps we can label the words or chunks for training.

Q3. Please provide the python code which can label a portion of text (maybe sentences) as insights. Please show the output of your code for the text attached as the input.

Ans:

1. We will assume that the text provided to us is corrected beforehand (**step 3 of Q2**).
2. We will use RNN(or GRU) to convert splitted sentences into single elements and tagging them.
3. Now using a pre-defined data set containing keywords that needs to be highlighted are first identified and then those words or chunks are highlighted.
4. The highlighted words or chunks are then compiled to form the summary.

We are using the python spacy library for POS recognition.

To extract nouns from the sentences, the attribute **.pos_** is used and extracted accordingly.

But if we want to extract the subject and the object from a sentence, we need to look at the relation between them and this is called Dependencies.

Each word is a node in the Dependency graph. The relationship between words is denoted by the edges.

The arrows carry a lot of significance here:

The arrowhead points to the words that are dependent on the word pointed by the origin of the arrow.

The former is referred to as the child node of the latter.

The word which has no incoming arrow is called the root node of the sentence.

References:

1. https://iq.opengenus.org/text-summarization-using-rnn/
2. https://towardsdatascience.com/pos-tagging-using-rnn-7f08a522f849