CSE538 NLP Assignment 4

Rushikesh Nalla
111971011

1 Introduction

Huge amounts of multimedia data is available over the internet via YouTube, MOOCs like edX, Coursera, Udacity etc. This is especially true at this point of time where most of the exciting content that is being produced is in the form of videos. The spoken part of these sources encodes very important information about the content. We can build a machine that listens to this audio data and extracts useful information to answer the questions posed to it.

Our project is a question answering system where the input is an audio story, a question and 4 option choices. The system produces the correct answer choice as output. Most of the QA systems focus on tasks where the answer to the question are straight forward (directly available in the text/data). For our project we are exploring TOEFL listening comprehension data that includes complex questions that ask for the conclusion of the theory, inferences etc presented to the listener. So, a clear understanding of the text is required for the system to answer these kinds of questions.

This task of machine comprehension is very challenging. Firstly, this system requires the spoken content to be first transcribed into text by ASR or manually, and the machine will subsequently process this output. ASR (though state of the art) still has a very high error rate. The amount of labelled data available for this problem space is very limited.

The approaches that have been explored can be divided into broadly two types - Non-neural and neural based. Non neural based approaches include:

- **Question/option similarity -** We now evaluate the cosine similarities between question and each of the option vector. The option with the highest similarity is deemed correct. The main disadvantage of this approach is that it ignores the story completely.

- **Sliding window -** This approach overcomes the shortcoming of the above approach by taking the story into account. In this approach we slide a window of fixed size and compare the cosine similarity of the window with the question word vector. The word vector of the window with maximum cosine similarity is compared with each of the choices. The option with the highest similarity is selected.

I am reviewing three of the neural based methods (Deep LSTM Reader, End-to-end memory network, Hierarchical Attention Model) that have been explored earlier in this problem space in the next section.

2 Paper Summary

2.1 Teaching Machines to Read and Comprehend (NIPS 2015)

2.1.1 Task Definition and Objective -

For developing a question answering machine we need a very large labelled dataset for training and testing the model. This was a limitation at that time (and it still is). They have created their own dataset of document, query and answer triples from CNN and Daily Mail websites which contain million news stories. The general idea is to use the summary points/abstract of news articles for coming up with questions/answers by replacing each entity at a time. Their task was to use this dataset and develop a question answering system.

They use the idea of entity replacement and permutation. All the entities in the document are replaced with some marker (entity1, entity2 etc) and it is permuted randomly each time a data point is loaded. This intuition makes sense because we want a corpora for evaluating the models ability to understand a single document (based on context) and not general knowledge.

They have described a couple baseline approaches but I will be discussing the main/neural (LSTM) approach.

2.1.2 Key/Technical Ideas -

Prior to this paper, neural models have been used in many NLP tasks. For this system we have to estimate the probability of word a (answer) from document d and question q,

$$p(a|d,q) \propto exp(W(a)g(d,q) \quad s.t.\ a \in V$$

where V is the vocabulary (it includes all the words in the documents, questions and the entity maskers), W(a) is the row a of weight matrix W, g(d,q) is the vector embedding of the document and question pair.

Deep LSTM can be used for translation as they have the ability to represent long sequences as a vector which contains enough information to generate a sentence in another language. Documents are usually very long and the idea is to use a Deep LSTM Reader. We can feed the document word by word then a delimiter (to separate document and question) and then the question can be feed word by word right after. We can also send the question first and then the document. The model will then process this pair as a single long sequence. We would then get a fixed vector embedding for the (document, question) pair. The model then predicts which word is the answer from the document.

The Deep LSTM cell has connections from each input to every hidden layer and each hidden layer to the output. The paper contains the standard LSTM equations with some modifications (sending document and query embedding together) and the Deep LSTM Reader is g(d, q).

- **Attentive Reader -** The Deep LSTM Reader suffers when the answer to the question is very far in the document. We need to have a mechanism to focus on which words of the document are relevant for the question at any point. They propose an attention model which first encodes the document and the question using separate bidirectional single layer LSTMs. The outputs of the forward and backward LSTM's are then concatenated. The Attentive Reader employs a token level attention mechanism where the tokens are embedded given their entire future and past context in the input document as we are using bidirectional LSTM.

- **Impatient Reader -** In this model as each question token is read, we read the document. At each token of the question the model computes a document representation vector using the bidirectional embedding. In this way we can get information from the document as we process each token of the question.

2.1.3 Conclusion and Thoughts -

They have provided the percentage of times the correct answer is one among the top N (1, 2, 3, 5, 10) entity markers in the document. The numbers clearly portrayed the difficulty in the task as the model cannot distinguish between entity marker and regular words.

The Attentive (69% accuracy) and Impatient Readers (68% accuracy) outperformed Deep LSTM Reader (62% accuracy) on Dailt Mail test data as expected because the documents are usually very long and attention is extremely important in this scenario (question answering). Even a single Layer Attention LSTM outperformed Deep-LSTMs. Given the length of document, a simple LSTM network didn't do that bad. The heatmap shown in the paper was a good visualization for the attention mechanism.

I really liked the approach of building the dataset in a cost effective fashion and the performance of simple LSTM for this task really shows its true capabilities. We have looked at such a QA application in class and this uses similar concepts.

This model fails when given complex questions i.e. the answer requires some complex inference which is necessary for the task (questions in TOEFL) we are tackling for our project. We need to develop a better model by maybe using a Tree-LSTM with attention that would help in better representing the syntactic and sematic structure of the document and question. Another way could be to use two different LSTMs/Tree-LSTMs with attention, one for document and the other for the question and look at ways for combining them.

2.2 End-To-End Memory Networks (NIPS 2015)

2.2.1 Task Definition and Objective -

This paper discusses a system for question answering which is trained end to end i.e. it learns all the features that occur between the input x and output y and requires less supervision during training. This model uses a notion of explicit storage and attention as needed when dealing with very long input sequence.

It is a novel RNN architecture where it reads the external memory multiple times before giving the output and has multiple computation hops for better performance.

This model differs from general RNN and LSTM because as it uses a global memory with shared read and write functions. As there are layer-wise weights in this model it can be viewed as a form of RNN which produces an output after a fixed number of time steps (number of hops), with the intermediary steps involving memory input/output operations that update the internal state.

The dataset used by the model had 20 different types of QA based on the form of reasoning and deduction. This network takes in a set of sentences (text) x and a query q as input and outputs the answer q (can contain multiple words). It writes all sentences to the memory up to a fixed buffer size and then finds a continuous representation for the x and q. This representation is then processed via multiple hops to output a.

2.2.2 Key/Technical Ideas -

The sentences have been encoded in two different ways -

- **Bag of words (BOW) representation -** It takes each word embedding of the sentence and sums the vectors $m_i = \sum_j Ax_{ij}$ and $c_i = \sum_j Cx_{ij}$. The question vector is also represented in a similar way. This cannot capture the order of words in a sentence (important of QA tasks).

- **Position Encoding -** Encodes the position in the sentence using $m_i = \sum_j l_j \odot Ax_{ij}$ where $\odot$ is element wise multiplication l_j is a column vector with the structure $l_{jk} = (1 - \frac{j}{J}) - \frac{k}{d}(1 - \frac{2j}{J})$, where J is number of words in the sentence and d is the dimension of the embedding.

Temporal Encoding - QA tasks also need to which sentence occurs before (temporal context). The memory vector is modified to $m_i = \sum_j Ax_{ij} + T_A(i)$ and the output embedding is modified to $c_i = \sum_j Cx_{ij} + T_C(i)$. $T_A(i)$ and $T_C(i)$ matrices are learnt while training.

Single Hop Model
Input memory representation - The input sentence $\{x_i\}$ is converted into memory vectors $\{m_i\}$ of dimension d. Represented in either of the two ways as described above. The query is also embedded in a similar way to get internal state u. We take the inner product of the two and apply softmax to get the probability vector.

$$p_i = softmax(u^T m_i)$$

Output memory representation - Each $\{x_i\}$ has a corresponding output vector $\{c_i\}$. The output memory vector o is:

$$o = \sum_i p_i c_i$$

.
Final Prediction - The sum of the output vector o and the input embedding u is then multiplied with weight matrix W and passed through softmax to produce the output:

$$\hat{a} = softmax(W(o + u))$$

Multi hop Model
The memory layers are stacked on top of each other. The input for layer k+1 is the sum of input and output vectors of layer k. Each layer has its own embedding matrices. They have a RNN like structure when the input and output embeddings are the same for each layer. They have used a linear mapping H to update u between hops, $u^{k+1} = Hu^k + o^k$.

2.2.3 Conclusion and Thoughts -

They have experimented a lot with the input (sentence representation, question type), parameters of the model (training methods, number of memory hops etc). They have compared the results with previous techniques as well.

When they have used the position encoding sentence representation, they have achieved better results as compared to Bag of words sentence representation. As the computation hops were increased, the performance also increased.

This neural network model with an explicit memory and a recurrent attention mechanism for reading the memory performed well in QA tasks. This model can also be used for language modelling and it performed better than RNN and LSTM. The idea of using external memory is quite exciting/intuitive looking at the performance of RNN and LSTM in various NLP tasks. Having multiple memory layers was a great extension.

This model fails on some complex QA tasks and not up to the level of memory networks trained under strong supervision. The task we are focusing for our project is complex and requires many attention hops.

2.3 Hierarchical Attention Model for Improved Machine Comprehension of Spoken Content (SLT 2016)

2.3.1 Task Definition and Objective -

The main idea is to develop machine which can automatically comprehend spoken content and summarize the key information from it. This can be extended to developing a question answering system.

Test of English as a Foreign Language (TOEFL) is a standardized test to measure the English language ability of non-native speakers wishing to enroll in English-speaking universities. In the listening section of the test, each question consists of an audio story, a question, and four answer choices. Among these choices, one or two of them are correct. Given the manual or automatic speech recognition (ASR) transcriptions of an audio story and a question, machine has to select the correct answer out of the four choices. This is the task at being tackled in this paper.

The questions posed in this exam are challenging for the machine as some of them require making inferences, finding gist and conclusions of stories. So we need a system which can comprehend such ideas easily. They propose a Hierarchical Attention Model (HAM) to construct tree-structured representations for sentences from their parsing trees and then estimate attention weights on different nodes of the hierarchies.

2.3.2 Key/Technical Ideas -

In this model, tree-structured long short-term memory networks (Tree-LSTM) are used to obtain the representations for the sentences and phrases in the audio stories, questions and answer choices. The memory module includes attention modules which give question-related attention weights for different sentences in the story. The answer module evaluates the confidence scores of the answer choices and gives the answer.

Tree-LSTM generates a vector representation for each node in the dependency tree based on the vector representations of its child nodes. It has similar computation (equations) of a standard LSTM. IN story module, the hidden vectors for all nodes in the tree structures of each sentence in the story are stored. The question module produces the hidden states of the root nodes, V_{S_i} of the Tree-LSTMs for the sentences S_i in the question. The question vector V_Q is computed as the sum of V_{S_i} for all S_i in a question.

The memory module extracts the information in the story relevant to the question V_Q based on representations obtained from the story module.

Attention Mechanism -

- **Phrase-level:** Let $O = o_1, o_2, ..., o_T$ be representations of the phrase (hidden states of Tree-LSTM).
- **Sentence-level:** Let $O = o_1, o_2, ..., o_T$ be representations of the sentences (hidden states of Tree-LSTM).

in both cases, these vectors are transformed into memory vectors $M = m_1, m_2, ..., m_T$ and evidence vectors $C = c1, c2, ..., c_T$ by the embedding matrices $W^{(m)}$ and $W^{(c)}$,

$$m_t = W^{(m)} o_t \quad , \quad c_t = W^{(c)} o_t$$

The question vector V_Q is similarly transformed to q_0.

$$q_0 = W^{(q)} V_Q$$

They have used cosine similarity to compute the attention score η_t between the query vector q_0 and each memory vector m_t, and then normalized it using softmax function.

$$\eta_t = q_0 \odot m_t \quad \alpha_t = \frac{\exp \eta_t}{\sum_{i=1}^{T} \exp \eta_t}$$

Each attention weight α_t corresponds to a memory vector m_t and an evidence vector c_t that represent a phrase or a sentence. The story vector s_o is calculated as the weighted sum of the evidence vectors c_t with the attention as weights.

2.3.3 Conclusion and Thoughts -

This model is one of the most mature one to carry out this task. Tree structured models help in representing the syntactic structure of the sentence and understanding the semantics. Attention and memory modules are essential when dealing with long sentences because for answering a question we are interested in finding out the most relevant words in the text and this can be achieved by having access to previous states/words of the story/question.

Most of the audios were converted to text manually and the rest were converted using ASR (obtained from CMU Sphinx Recognizer). They have compared their results with Deep LSTM Reader, MEM2NN models (which I have summarized earlier) and some other models as well. As this is a much harder problem in terms of question complexity, Deep LSTM Reader failed badly. This model has achieved much better accuracies (on both manual (49%) and ASR (48.8%) transcripts) compared to all other previously known methods.

They have analyzed HAM model with phrase/sentence level attention mechanism with 1 and 2 hops. Phrase level attention with 2 hops performed best which might be because the first hop selected the key phrases and for the second hop the attention was changed based on first hop results.

The questions were divided into 3 types: basic comprehension (Type 1), understanding attitude of speaker (Type 2) and inferences/conclusions (Type 3). This was also compared across various models and HAM model was able to beat other models significantly when answering Type 3 questions (difficult ones).

The idea of using Tree-LSTM with hierarchical attention is really clever as it captures the sentence structure as well as the information needed for answering a question. Improving this model is possible by increasing the dataset as it is quite limited. Because of this limitation, We can maybe try using transfer learning (from other spoken content) but how effective it would really be has to be evaluated.

3 Discussion

The papers that I have summarized are all related to the task of building question answering systems. All of them proposed architectures of varied nature but suitable to this task. I now have a clear idea about how to approach a problem (learning different thought processes). The major shortcomings of Deep LSTM Reader and MEM2NN are they cannot answer complicated questions. They somehow expect the answer to be present in the story in a straight forward manner.

The HAM architecture was developed by combining two simple ideas of using tree LSTM and attention mechanism (both are important for QA tasks) and hence produced powerful results. We look to implement MEM2NN and HAM architectures as proposed and try to match the accuracies they have obtained if not increase.

We might extend HAM and use a multiple layer attention mechanism as seen in MEM2NN which could improve the results. We will try and tweak the architecture by changing a bunch of parameters and do a qualitative analysis. Using transfer learning might also help but given the time constraint it will be difficult to implement right now. Overall working on this project (QA system) is exciting and I have learnt a lot in this space.