Allowing the Deaf to Hear: Natural language processing utilizing tree-structured neural networks

Shivam Syal

Middlesex County Academy of Science, Mathematics and Engineering Technologies, Edison, NJ

Table of Contents

Abstract

Title: Allowing the Deaf to Hear: Natural language processing utilizing tree-structured neural networks

Name: Shivam Syal

School: Middlesex County Academy of Science, Mathematics and Engineering Technologies, Edison, NJ

Teacher: Ms. Courtney Macdonald

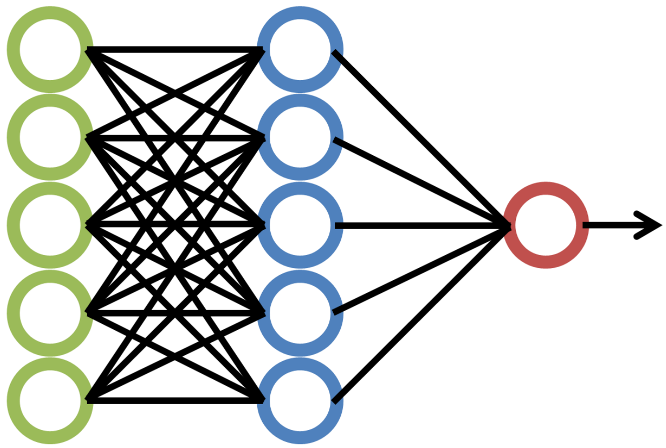
NATURAL LANGUAGE PROCESSING is a progressing branch in the Artificial Intelligence field, proving to exploit valuable input parse information as they interpret the meanings of sentences with the use of tree-structured neural networks. Multitudes of these traditional systems have been implemented in applications for the deaf to better understand the world around them. However, tree-structured neural networks are exponentially slowing down and require pre-parsed inputs which increases the time delay of outputs and decreases the efficiency of the system overall. Through numerous test trials, dynamic inputs, and white papers, my research develops upon a new method of tree-structures neural networks: hybrid networks. Instead of having to pre-parse input data and interpretation in two seperate tree-structures, this hybrid model will allow for a single tree-structure for both processes. This minimizes the processing time and power required as a hybrid tree-structure combines the standard parser and interpreter. As a result, each NLP model can be executed exceedingly faster than a traditional tree-structure. Implementations for this method are endless; the deaf will be able to communicate more

efficiently and quickly, foreign language translation, increased communication for the disabled (cerebral palsy, dysgraphia, etc.) and more. Not only that, but less CPU power, battery life, and memory will be exploited with a singular branch for each NLP model. This research paper elaborates on the ongoing investigation, implementation, and future work of the hybrid tree-structures for natural language processing.

1 Introduction

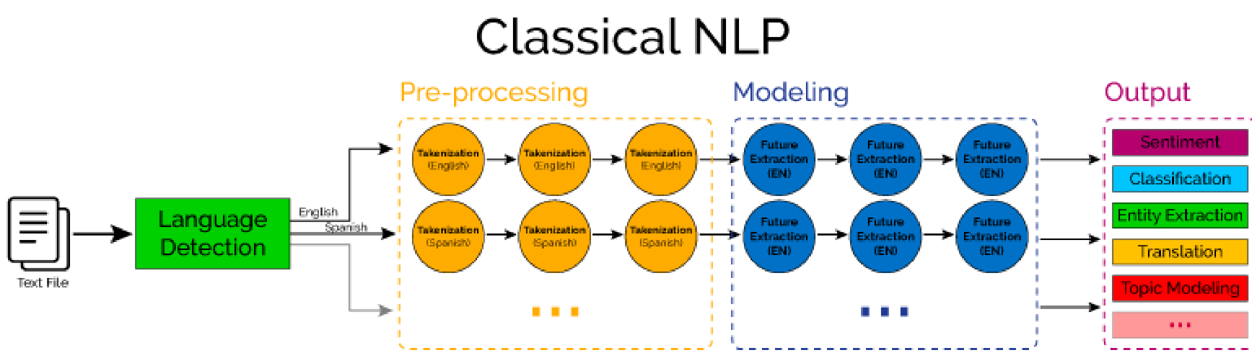
In the world of Artificial Intelligence, the brilliant minds of our exponentially progressing technological field have created a variety of specialties in the areas of machine and deep learning. From the creation of simple two layer neural networks, to giving computers the ability to think and reason, the technologists of today are completely changing the face of the future. One of these areas is Natural Language Processing. Natural Language Processing, or NLP, has found its way into our daily lives. Utilizing input data such as words, frequencies, voices, or tones, NLP has advanced to a stage where it can create sentences of meaning, songs with harmony, and near to real-life voices. These applications are implemented in personal assistants like Siri or Alexa, text editors like Word or Google Docs, search engines like Bing and Google, and more. The question that initially must be answered though, is, how does NLP manage to perform these complicated tasks from mere input data? To answer this, we must understand the realm of the key buzzwords in modern technology: Artificial Intelligence, Machine Learning, and Deep Learning.

Artificial Intelligence or AI is the first key to our NLP puzzle. Patrick Winston, the Ford professor of artificial intelligence and computer science at MIT, defines AI as "algorithms enabled by constraints, exposed by representations that support models targeted at loops that tie thinking, perception and action together." At its core, AI is the branch of computer science that aims to answer Turing's question in the affirmative. It is the endeavor to replicate or simulate human intelligence in machines. This is done through a series of models, neural networks, and deep learning. Deep learning is a subset of machine learning that employs artificial neural networks that learn by processing data. Artificial neural networks mimic the biological neural networks in the human brain. Multiple layers of artificial neural networks work together to determine a single output from many inputs, for example, identifying the image of a face from a mosaic of tiles. The machines learn through positive and negative reinforcement of the tasks they carry out, which requires constant processing and reinforcement to progress.



Model 1: Three layer deep learning neural network

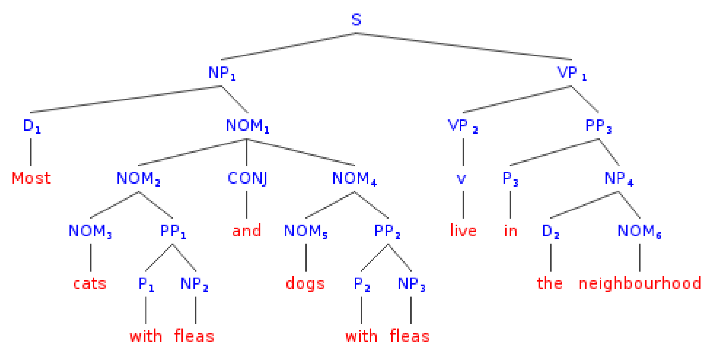
As seen in Model 1, a neural network is composed of many neurons. In this case, each neuron contains a weight to its input, and a bias. This will help control the final output, as the specific weight and bias on each neuron determines what will happen in the next layer, and the next one, and so on so forth. Now, in terms of natural language processing, this input could be a specific letter, a frequency, a word, a sentence, etc. This input would be assigned a weight and bias. Based on this, the neural network will determine which neuron is activated.



Model 2: Classical NLP neural network model

1.1 NLP neural network and tree-structures

With the classical NLP neural network in Model 2, there are two main sections: pre-processing, or parsing; and modeling, or interpretation. These two steps in the process of an NLP operation take specific amounts of time, and based on their sizes, affect the efficiency of a system variably. In contrast to a traditional deep learning neural network, the NLP model provides for processes based on application, such as translation, classification, and others. These variables also, in turn, affect system efficiency. For both the parsing and interpretation, there are different tree-structures. These are intricate and branched out structures used in NLP models to both parse inputs, and in turn, after fully parsing all sections of the input, can interpret its meaning based on previous training modules. This process is used in the current TreeRNN implementation as described in Let us take the sentence, *Most cats and dogs with fleas live in the neighborhood.* This sentence can be broken down into different sections, as shown below in Model 3.



Model 3: NLP tree structure for pre-processing input

Because TreeRNNs use a different model structure for each sentence, efficient batching is impossible in standard implementations. Partly to address efficiency problems, standard TreeRNN models commonly only operate on sentences that have already been processed by a syntactic parser, which slows and complicates the use of these models at test time for most applications. **[1]**

1.2 What is a hybrid tree-structure

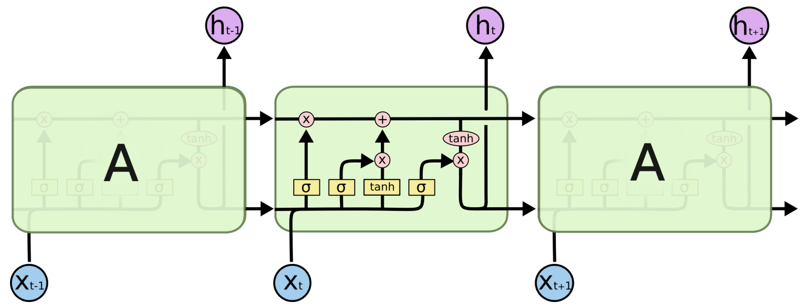
Based on major flaws with the TreeRNN structures in terms of efficiency, processing power, and extra complications, the use of a different method is not only favorable, but somewhat necessary to decrease the run time of applications, and maximize CPU usage. As a result of many recent research papers and test trials, technologists have come together to formulate a new method of NLP modeling training and execution: the hybrid tree-structure neural network. This type of neural network “executes the computations of a tree-structured model in a linearized sequence, and can incorporate a neural network parser that produces the required parse structure on the fly.” With clocked speeds of application run-time being almost 25 times faster than traditional TreeRNNs, it can simultaneously parse and interpret unparsed sentences, removing the dependence on an external parser at nearly no additional computational cost. **[1]**

2 Methods and Materials

The tree-structured neural network has been widely used as natural language processing parse structures. These NNs are referred to as recursive neural networks. Recursive neural models (also referred to as tree models), by contrast, are structured by syntactic parse trees. Instead of considering tokens sequentially, recursive models combine neighbors based on the recursive structure of parse trees, starting from the leaves and proceeding recursively in a bottom-up fashion until the root of the parse tree is reached. Standard recursive/Tree models work in a similar way, but processing neighboring words by parse tree order rather than sequence order. It computes a representation for each parent node based on its immediate children recursively in a bottom-up fashion until reaching the root of the tree. For a given node η in the tree and its left child η_{left} (with representation e_{left}) and right child η_{right} (with representation e_{right}), the standard recursive network calculates e_η as follows: **[4]**

$$e_\eta = f(W \cdot e_{\eta_{\text{left}}} + V \cdot e_{\eta_{\text{right}}})$$

In spite of this method, in a paper by Hochreiter & Schmidhuber (1997), a new NN was introduced: Long Short Term Memory networks. Usually just called “LSTMs”, these are a special kind of RNN, capable of learning long-term dependencies. All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer. LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.



Model 4: The repeating module in an LSTM contains four interacting layers. [2]

In Model 4, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations. The usage of LSTMs has produced promising results, and have started to become more mainstream in NLP parsing and interpretation tree-structures. Now what hybrid tree-structures provide is for a combination of both of these LSTMs for tree-structures and simultaneously give us processed outputs without having to use an external parser for unparsed data.

3 Experiments and Results

To test the proposed method, we consider the following tasks, each representative of a different class of NLP tasks:

- **Sentence-Target Matching** with the use of a question answering machine, which are parse tree nodes for recursive models and different time-steps for recurrent models.
- **Speech Recognition** to further analyze the effects of RNNs, LSTMs and hybrid tree-networks on NLP models

3.1 Sentence-Target Matching [4]

Task Description: In the question-answering dataset QANTA, each answer is a token or short phrase. The task is different from standard generation focused QA task but formalized as a multiclass classification task that matches a source question with a candidates phrase from a predefined pool of candidate phrases. For example:

Question: *He left unfinished a novel whose title character forges his father's signature to get out of school and avoided the draft by feigning desire to join. Name this German author of The Magic Mountain and Death in Venice.*

Answer: *Thomas Mann* from the pool of phrases. Other candidates might include George Washington, Charlie Chaplin, etc.

The model of Iyyer et al. (2014) minimizes the distances between answer embeddings and node embeddings along the parse tree of the question. At test time, the model chooses the answer (from the set of candidates) that gives the lowest loss score. As can be seen from the results presented in Model 5, the difference is only significant for the LSTM setting between the tree model and the sequence model; no significant difference is observed for other settings.

	Standard	LSTM
Tree	0.523	0.558
Sequence	0.525	0.546

Model 5: Test set accuracies for UMD-QA dataset.

Discussion The UMD-QA task represents a group of situations where because we have insufficient supervision about matching (it's hard to know which node in the parse tree or which time step provides the most direct evidence for the answer), decisions have to be made by looking at and iterating over all subunits (all nodes in parse trees or timesteps). The results above illustrate that for tasks where I tried to align the target with different source components (i.e., parse tree nodes for tree models and different time steps for sequence models), components from sequence models are able to embed important information, despite the fact that sequence model components are just sentence fragments and hence usually not linguistically meaningful components in the way that parse tree constituents are.

3.2 Speech Recognition

Task Description: Using the RAVDESS [3] dataset, a set of audio samples containing sentences and words with varying emotions were used to provide separate moods for the interpreter. The experiment was implemented in jsPsych (de Leeuw, 2015) and psiTurk (Gureckis et al., 2015), along with custom code for sending audio data to the Google Cloud Speech API.

Use of GCS: I chose the Google Cloud Speech API due to its ease of use, the ability to provide a "speech context" (which played an important role in improving the transcription accuracy), the ability to obtain confidence ratings for each transcribed utterance, and the ability to automatically identify vocalization onset times. Not only that, but it utilizes tree-structured NNs, which helps in terms of fast-data capture with jsPsych and RNN capabilities.

Usage: This application of NLP models using tree-structures can be helpful not only to make our machines smarter, but for the benefit of the deaf, as this speech recognition can help the deaf understand the world around them; increased communication for the disabled (cerebral palsy, dysgraphia, etc.); help foreign exchange students understand an unknown language in the area they are living in, and translate their surrounding voices in real-time; and more.

Results: The above findings show that human speech-to-text transcription, recovers many of the fundamental behavioral phenomena in free recall data. Our results provide a proof of concept that automatic speech-to-text transcription is sufficiently accurate to serve as an effective substitute for human annotators in list-learning experiments.

4 Discussion and Conclusion

With the presentation of various experiments and results, we must address this research paper's end goal; with the heightened advantage of hybrid tree-structures, we can build applications that can help the deaf understand the world around them with more speed, and more efficiency. If programs start and continue to use this method, we can essentially allow the deaf to hear with the capabilities of new and improved NLP models that can quickly parse and process language inputs and provide textual outputs. Not only the deaf, but people with other disabilities which do not allow them to fully understand the world around them. Apps can be created to help them increase their audible potential. In addition, NLP can provide access to education for students around the world with English based lectures being able to be translated in real-time for foreign students. NLP with the utilization of tree-structures provides the world and numerous groups of individuals with the opportunities they might have never had. This paper applies existing models to existing tasks, barely offering novel algorithms or tasks.

My conclusion can be summarized as follows:

- Recurrent neural networks have proven effective in the past few years of NLP growth with the implementation of TreeRNNs, and now with the usage of LSTMs. However, these methods have proven inefficient and require lots of pre-parsed and pre-processed inputs/conditions. Removing these restrictions can help us create applications for the disabled.
- Despite the fact that components (outputs from different time steps) in recurrent models are not linguistically meaningful, they may do as well as linguistically meaningful phrases (represented by parse tree nodes) in embedding informative evidence, as demonstrated in QANTA task. The experiments detail that recurrent models like LSTMs can discover implicit recursive compositional structure.

The use of tree-structured neural networks and hybrid tree-structures models will help us bring words to text faster and more efficiently. It can help allow the deaf to theoretically hear.

4 References

1. https://arxiv.org/pdf/1603.06021.pdf
2. https://colah.github.io/posts/2015-08-Understanding-LSTMs/
3. https://link.springer.com/article/10.3758/s13428-018-1037-4#Sec16
4. https://nlp.stanford.edu/pubs/emnlp2015_2_jiwei.pdf
5. https://pdfs.semanticscholar.org/4782/7d171277597caff30fb04d95ff6f162a984c.pdf
6. https://towardsdatascience.com/the-fall-of-rnn-lstm-2d1594c74ce0