[[1]](#footnote-1)

Topic Identification Using Back Propagating Neural Network

Osvaldo Rosado, Daniel Campos, Dwight Naylor

***Abstract— Natural Language Processing systems are widespread and commonplace, in some cases a lightweight system is much more useful and called for than robust enormous systems.***

# INTRODUCTION

There are many possible environments where people are interacting through natural language with a system. Many of these environments only need some kind of basic system understanding in order to function. Of these systems, most do not need an entire NLP suite, and by having one, the system is more prone to failure and harder to set up. With our topic identification system only a relatively small amount of training data is needed produce usable output to route further system processing. For our system, we used a small training set(~300 phrases to 7 categories) to produce a lightweight neural network that would be able to accurately determine the intent of a given question, so that the rest of our ‘Assistant’(an analog for voice recognition phone systems such as Siri, Google Now, and Cortana) system can properly answer the question. The ‘Assistant’ system then took this topic identification and passed the query along to the appropriate query channel.

Another possible way that a system may be used would be for basic Frequently Asked Questions on a website or simple customer service. A user would train the system on a small amount of data allowing the neural network to point the connected system to the ‘type’ of its query.

# Goal

Our goal was to create a natural language question parser. This system would be able to take in a question as input and output what category (topic) the question belongs too. For our system, we take in a question and we return what service our ‘Assistant’ software should query for the correct answer.

One of the primary uses of this tool would be to replace the current systems in ‘Assistant’, a regular expression parser. We replaced this system because this regular expression caused mishaps by giving tremendous weights to certain categories. For example, if a user were to ask “What is the definition of weather?”, the regular expression engine would map this query to a weather related query and return current weather. With the lightweight NLP system, ‘Assistant’ would have returned that said query was a encyclopedia query.

The primary reason for using this to replace the regular expressions is to add robustness to the system and to increase usability. By having a system that can act much closer to user speak (the training solely relies on the inputs) it means that users do not need to learn how to “speak” to the system and untrained users can use the system without frustration, much like a real-life assistant.

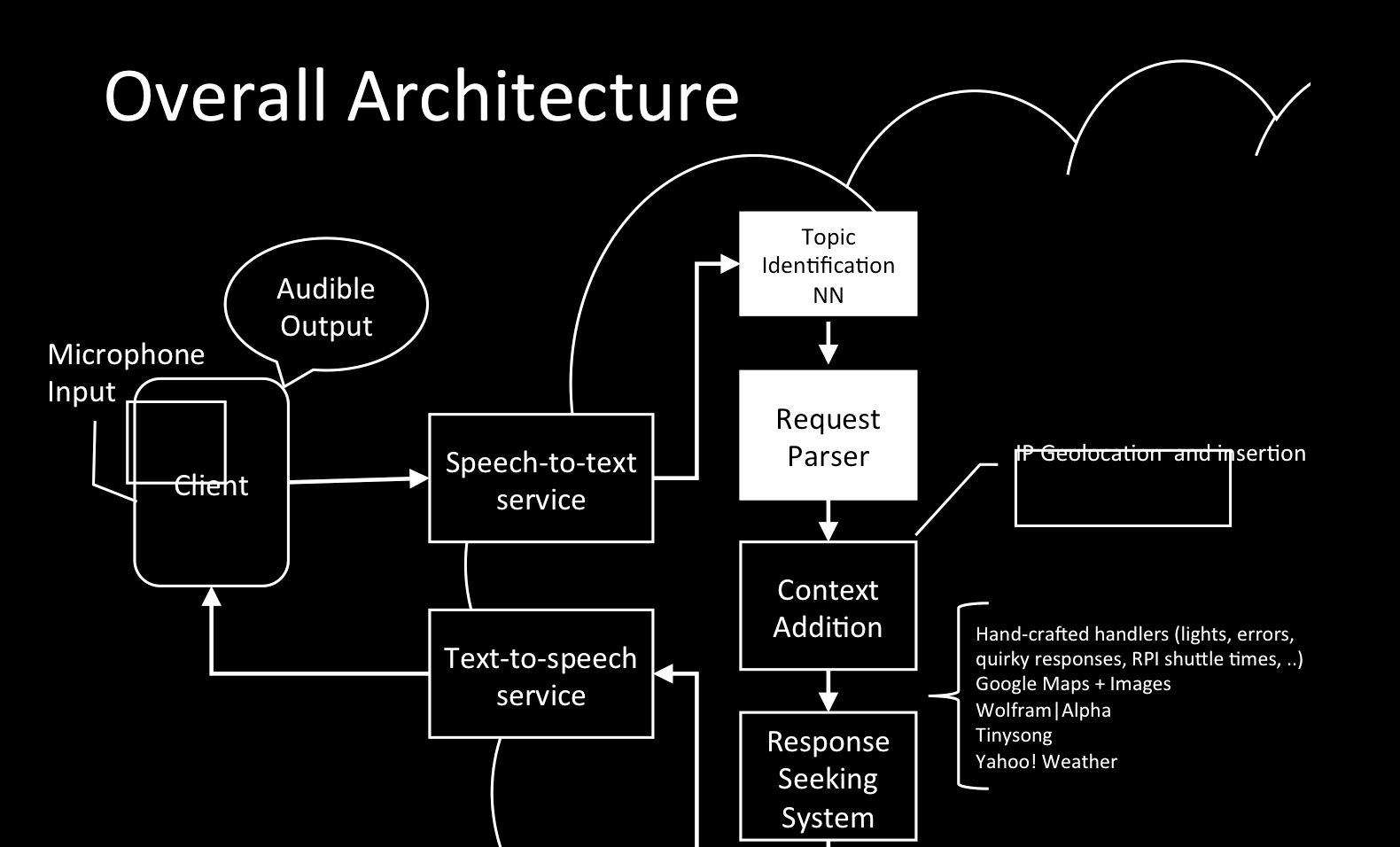
# Background and Motivation

We presently have an existing general-purpose digital assistant that can respond to free-form requests and perform actions across a wide variety of domains such as playing music, looking up encyclopedic questions, finding locations on a map, performing math and obtaining graphs, finding pictures on the web, and more.

However, this system currently relies on a long list of regular expression “templates” to match given input against. This is very fragile in real-life usage. For example, one can ask this system “What is the meaning of life?” and the system will understand the request and answer appropriately, but the system will not handle the very similar “Can you tell me the meaning of life?” despite it clearly being possible for the system to obtain the answer to what this question is asking for. Even more troubling, this system can correctly answer, “What does the word university mean?” as a word definition question but answers “What does the word weather mean?” as if the question was “What is the weather currently?” Very obviously, this system of template matching is insufficient for real-life usage.

With improving this assistant as our motivation, we set our problem to address as replacing this template matching system with the more robust natural language question parser that we have described in the Goal section. This problem is worth solving as the end result will allow for more useful machines. The presently existing assistants are very limited in scope and not open, where as a more general-purpose system like the one we seek here can be useful in many more domains with the potential for user customizability. Some projects already exist to solve this customizability issue, such as Jasper (<http://jasperproject.github.io/>) but these rely on phrase matching, much like our current system. A general-purpose assistant that has true natural language processing at its core will be a far superior solution.

# System Architecture and Approach



Our overall Assistant architecture follows a traditional cloud service model. A client running on a device listens continually for an audio cue to begin recording. Upon hearing this cue the audio is recorded until a sufficient amount of silence is found to indicate the end of the phrase. The utterance from the audio is then converted to text from a third-party system, presently the Google Speech API, and then this text is sent to our Assistant Cloud Service. The Cloud Service then responds with the natural language that the device should “speak.” The cloud service does processing in 3 stages.

The first stage, the Request Parser, which was previously a regular expression parser, is now a neural network. The system works by creating a neural network that can understand inputs similar to a collection of words together. For our purposed, we have called it an ‘iterative neural network’, in other literature it is referred to as a ‘back propagating neural network’.

For those unfamiliar with the jargon, a neuron (or perceptron) can be viewed as a machine, which takes a series of inputs, and calculates a weighted (often linear) sum to return as a result. A neural network is nothing more than a group of these neurons (dubbed "input neurons"), which may repeatedly feed into another group of input neurons (dubbed a "hidden layer"), eventually feeding into a final group of neurons from which output is discerned (dubbed "output neurons"). To train such a network, a sample input is fed into the network, at which point the calculations made by each neuron propagate through the network, eventually reaching the output neurons and calculating a result. This result is then compared to the sample output, and errors are calculated. These errors can then be back-propagated through the network, adjusting the weights of individual neurons to closer-approximate the given solution to the problem. Repeated enough times with a proper network, this will eventually yield a configuration capable of yielding the correct answer to all example inputs and outputs. Note that this network described is known as a "feed forward neural network" or FFNN, which is only one of many types of neural networks used.

This backwards-propagating neural network is a special type of FFNN, which operates by processing input one token at a time (for example, one word in a sentence at a time). Each token is fed into the network, and the network is tasked with correctly identifying important abstract structures as they appear within the sentence. This approach can be augmented by considering other data related to the token, such as neighboring tokens and position in the sentence, but the core concepts are most easily applied when limited to token appearance, and so this document will assume that is all the input being given (although the actual program will use these augmenting features).

One benefit of this approach is that it allows us to process a sentence of any size and structure, so long as we have a neuron associated with each possible input token in our input layer, and a neuron associated with each possible desired output in our output layer.

For out system we created hand-tagged sentences to represent the different topics that ‘Assistant’ used and tried to do so with an even distribution per for each category. The most important part of this program is that it can be re-adjusted for new datasets without too much difficulty, and still give useful results from any new dataset.

The next stage comes in the interpretation of the

the results. Our network has an output layer of N nodes where N represents the amount of topics that our results can have. Each one of these nodes has a value from 0 to 1 representing how close the output is to said category. A sentence is assigned a topic by whatever output node has a value closest to 1.

Next, this information moves on to the Context Addition stage where the client’s location is determined, through IP geolocation, as well as the current date and time, so that this information is available in the request as well. The final step is our Response Seeking System where we route the given question to an appropriate knowledge base based upon the type of question that was sent, formatted in a way appropriate for the selected knowledge base. When we get a response back from the knowledge base, it is converted into a reasonable natural language response and given back to the client to speak to the end-user. The first stage here, the Request Parser, is what we seek to replace and improve upon.

Much of the architecture for the replacement Request Parser has yet to be determined as good ways to move forward are still being researched. However, it is likely that the replacement parser will look for patterns in the construction of the given sentence, rather than patterns in the actual words being used. Part-of-speech tagging is likely to be of importance here to determine the actual topic of the given question.

Principal tasks in developing the replacement parser are difficult to determine at this stage, but some tasks are known. Researching the best technology to use for the parser will need to be done over a span of at most one week in order to have this project complete by December. Getting the parser to a state where it can properly determine the “type” of requests somewhat reliably will have approximately two weeks, and getting the “topic” of requests determined well will have another two weeks after that. The remaining time will be for integrating the parser back into the overall Assistant Cloud Service.

# Data And Analysis

During our project scoping we assemble a list of phrases that we believed were big ‘hiccups’ and that ‘assistant’ would hopefully get right with the neural network. We assembled a list of phrases including:

* When was Albert Einstein born
* Take me home
* Find me a image of a cat
* Find me a picture of a cat
* Show me a cat
* How old is Danny Glover
* I want to hear Bonfire

That totaled to about 120 entities. We then ran these verbal tests with the new ‘assistant’ web technology and categorized a pass or fail on each test if the system recognized the correct category. Prior to the new system assistant got all of these phrases wrong. With the neural network at the time of testing we got around 63% of the hiccups resulting in the proper category. While this is no where near the 100% coverage we want, we are confident with our future work we can reach this level or at least train a system that will avoid our these major ‘bad press’ moments.

# Work going forward

As we continue to work on our assistant we have identified a few areas of primary interest: Improve sentence preprocessing, creating a smaller and quicker neural network, creating more data and experimenting with data size, and improving the Speech to Text system.

The highest area of interest and perhaps the most useful is the improvement of our sentence preprocessing. We plan to add varying degrees of intensity to our sentence preprocessing beginning with removal of plurals and standardizing verb tenses. As we continue to develop our sentence simplification we add features such as determinant removal. We hope to be able to produce the simplest and most standard inputs that we can. As we produce the simplest input there will be less variation in both our training data and in our user input. We would also have to develop tools to make sure that sentences do not simplify that produce different results. This will hopefully reduce the need for a larger training set and will make out network smaller, thus fixing our other issues.

1. [↑](#footnote-ref-1)