Biologically Motivated Approaches to Speech Recognition

Dept. of CIS - Senior Design 2010-2011

Mishal Awadah
emish@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

Robert Hass
hassr@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

John P. Mayer
jpmayer@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

Dr. Mark Liberman
myl@ling.upenn.ed
Univ. of Pennsylvan
Philadelphia, PA

ABSTRACT

Speech perception is uniquely human, so applying theories about human psycholinguistic systems to computer speech recognition may improve performance. In the natural model, computations are performed by brains using data from ears. Thus, perhaps effective speech recognition can be performed by a brain-like machine learning algorithm receiving ear-like inputs. It is theorized that human listeners translate speech sounds into the lip and tongue movements that produced them - perhaps a speech recognizer that could relate sound to articulation would perform well. This project aims to employ Hierarchical Temporal Memory (HTM), a brain-like algorithm, in identifying ear-transformed audio with the help of X-ray articulation data. An experimental architecture was developed, but reliable HTM results were not obtained.

1. INTRODUCTION

A phoneme is the audio equivalent of a letter - it is the smallest perceptual unit of speech. The central problem of speech recognition is to identify the sequence of phonemes in an audio sample.

The machine learning algorithms generally applied to this problem, like as the Hidden Markov Model, do not exploit many plausible assumptions about sound data [10] and are thus general in scope and application. On the other hand, Hierarchical Temporal Memory networks (HTMs) take for granted that their inputs are explained by low-to-high-level 'causes', which they then attempt to identify [2]. That is, they posit that their input consists of patterns at various scales in space and time. This is a useful approach because speech recordings in fact possess this structure - there are multiple levels of causative factors that inform which sound a speaker produces, among which is which phoneme he or she is saying.

In terms of the ear's representation of sound, only some frequencies must be captured - ears only pick up sound between 20 Hz and 20,000 Hz. Nerves represent sound as an array of amplitudes in a sequence of frequency bins, with the number of frequencies in a bin approximately log-proportional to the frequency it captures. Various transformations of sound exist that capture some or all of these properties, as well as other ways to represent sound data that are useful for other reasons.

2. BACKGROUND

2.1 Speech

Humans perceive speech as a sequence of discrete phonemes but this is not reflected in sound waves. Audio recordings of speech bear little resemblance to common experiences of verbal or written language, and computer-based methods correlating the two have limited success. In addition to being poorly represented in sound waves, phonemes are pronounced differently based on which other sounds are around them. For example, the phoneme at the end of "books" is an 's'; the last phoneme in "bugs" is best described as 'z'. Representations of the physical sound in these two cases will be quite different, but they are perceived the same. The Motor Theory of Speech Perception offers an explanation for this [7] - it states that the sound is translated by listeners into the motions of the lips and tongue that produced it. The parts of a speaker's brain that control his articulators activate in a certain sequence as he says a certain thing; the Motor Theory of Perception holds that sound is processed by a listener into a representation of what his own brain would do to produce the same speech. This would mean that a very similar sequence of brain activity takes place in the listener as in the speaker, which would make speech perception almost exactly the reverse of the production process (as opposed to one being about moving muscles and the other about interpreting sound). An English speaker's articulators do very similar things to produce 'z' as opposed to 's', which may explain why they are perceived interchangeably despite sounding different. Preliminary brain imaging studies support this hypothesis, but the evidence is far from conclusive. It has been shown that the regions of the brain controlling a listener's lips and tongue, respectively, activate upon hearing 'p' and 't'[1] [12].

From a computational perspective, it makes sense that inferring articulator motions from sound might be plausible because speech sounds are produced in the first place by manipulation of articulators. It also seems likely that correlating sequences of articulator motions to words would be somewhat straightforward, because this is relatively invariant from one speaker to another. In short, there is strong theory but little data to support a place for articulators in speech recognition between sounds and word. Regrettably, articulators were not ultimately incorporated into this project.

Many databases of speech are available to researchers working on various problems. Among the most exotic is the University of Wisconsin X-ray Microbeam Speech Production Database (μDBD), which contains data of articulatory

[1]As an English speaker can verify, 'p' is produced exclusively with the lips and 't' exclusively with the tongue.

positions alongside an orthographically[2] transcribed audio recording. This combination of articulators and speech data could be used to examine the relationship between the two phenomena.

The most popular sound database is the Texas Instruments/Massachusetts Institute of Technology (TIMIT) corpus of read speech [1]. It includes recordings of 630 speakers from 8 major dialects of American English reading sentences with a wide range of phonemes. The corpus contains orthographic transcriptions and time-aligned, annotated phonemes. TIMIT is the standard database for phoneme research; it was used by many of the studies related to this project, and results from this database are the most useful metric for comparison with existing work.

In contrast to TIMIT's complexity, the Hillenbrand vowel database is a collection of single-vowel utterances. It contains twelve vowel classes, and draws from approximately 200 speakers of varying age and gender. All vowels are in identical context (they are preceded by 'h' and followed by 'd'). The uniform pronunciation of each utterance greatly simplifies classification. It is this database that was used for the current project.

2.2 Hierarchical Temporal Memory

Hierarchical Temporal Memory (HTM) is a recent machine learning model developed by Numenta, Inc., that replicates the structural and algorithmic properties of the neocortex [4]. This is relevant as the neocortex is the part of the human brain responsible for higher level functions such as sensory perception, spatial reasoning, and language. An HTM network attempts to identify hierarchical patterns in its input in space and time – high-level phenomena are referred to as *causes* of their lower-level instantiations.

HTMs' tiered structure accounts for their preference for topologically-organized inputs. Data from relatively small regions of perceptual space contain patterns that become increasingly abstract (and broad in area) in higher levels of the hierarchy [4]. This arrangement is similar to that found in human vision and audition.

HTMs are organized as a tree-shaped hierarchy of nodes, where each node implements a simple learning algorithm. Within an HTM, each node of the bottom level will look at only a small subset of the incoming data streams. Streams will first be grouped together by a single frequency band. At higher levels, larger sets of frequency bands will be aggregated, with the root node experiencing the effects of all of the input streams together with the overall energy waveform. This way, lower levels of the hierarchy will group together fast, local patterns, where upper levels will group together slower and more global patterns. The output of the top node of the HTM will be a classification of an instance of a particular phonetic event.

The input to every node, regardless of its position in the hierarchy, is a temporal sequence of patterns in vector form. It can be said that each node is receiving a "movie" of patterns as input. Each node contains two components referred to as "poolers". The spatial pooler maps incoming vector data to currently stored vectors that have already been seen, and the temporal pooler groups vectors together based on how close they occur to each other in time.

Structure of a node: A node has a spatial pooler and a temporal pooler. The input to the node is connected to the input of the spatial pooler. The output of the spatial pooler is the input to temporal pooler. The output of the temporal pooler is the output of the node. The node shown in figure 1 has an input pattern of size 6. The spatial pooler within this node has 5 quantization centers marked c1 to c5. The output of the spatial pooler is a vector of length 5. The temporal pooler within this node has 2 groups marked g1 and g2. The output of the temporal pooler, and hence of the node, is a vector of size 2 [2].

[2]Plain-text transcription, as opposed to phonemic transcription.

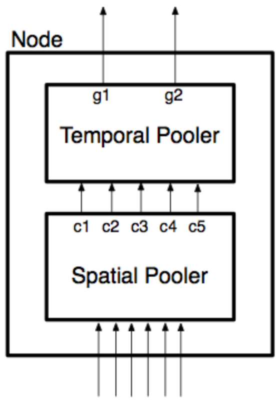

Figure 1: HTM Node Detail

Every node, and as a result the HTM as a whole, operates in two modes: training and inference. A node in training mode activates its spatial pooler first, and then trains its temporal pooler using the inferences of the spatial pooler. Incoming vector data is grouped into quantization points based on a similarity measure. These points are defined by setting a Euclidean distance D which the spatial pooler uses to compare currently stored quantization points to the incoming data vectors. If incoming data differs from currently stored data by at least D, then a new quantization point is created to represent that data. Otherwise, no new data is added to the spatial pooler. Once this operation has gone on for long enough such that quantization points are no longer being added, or the pooler has reached an acceptable threshold, the temporal pooler is activated.

The temporal pooler identifies patterns in time, and so begins to group quantization points together based on their temporal proximity. As new vector data is received by the node, quantization points are outputted from the spatial pooler in a sequential manner. So if event A is continuously followed by event B, for instance, the temporal pooler will create a group that contains A followed by B. Ignoring much of the complicated logic governing this learning scheme; the temporal pooler creates a first-order time-adjacency matrix where time coherent group creation can be viewed as finding the most highly connected sub-graphs from the graph represented by the adjacency matrix [2]. Once these temporal transitions are learned and corresponding groups of centers created, the temporal pooler starts producing output in terms of its learned temporal groups. The output is a vector of size equivalent to the number of temporal groups created, and can be considered as a probability distribution over the space of the temporal groups [2].

The operation of nodes in the hierarchy follows a level

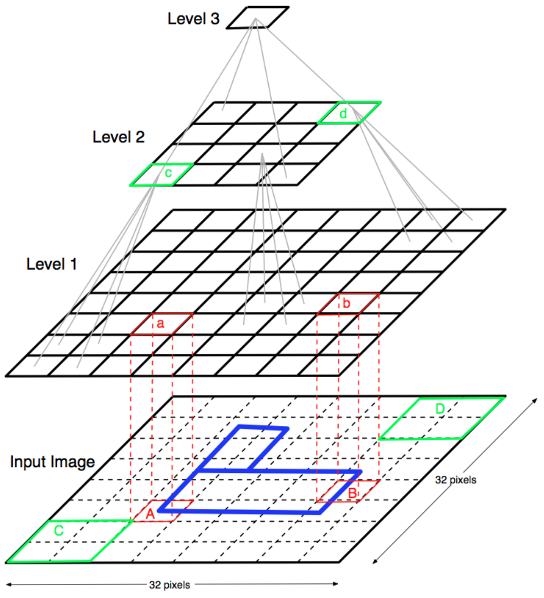

Figure 2: HTM Vision Example

by level strategy. Preliminarily, nodes at the bottom level are trained while the rest of the nodes are disabled. Once all the nodes at level i are finished learning, they begin to output to level $i+1$ nodes. The level i nodes are said to have reached the "inference" stage, and the level $i+1$ nodes are enabled and are now in the "learning" stage. This process repeats until all nodes are fully trained. When this finally occurs, the HTM as a whole can begin inferring high-level causes of its input data. The output of the HTM is the likelihood of each of the highest level causes identified in its the sensory inputs. Figure 2, for example, shows how an HTM classifies images. The input image is fed to level 1 of the HTM in a 1 to 1 mapping, and later inferences are propagated to higher layers in a 4 to 1 mapping. The higher layers are thus determining the causes that affect change in the bottom layers they receive input from. It is in this sense that HTMs employ a hierarchical learning mechanism.

The HTM framework is promising because of the hierarchical "cause and effect" structure it represents, which can be used to identify phonemes as causes for certain articulatory movements associated with them, and on a sub-level, identify certain articulatory movements as causes for certain auditory properties. Furthermore, both speech and articulatory movements are spatial patterns occurring through time, despite speech patterns being represented by transformations. This bodes well with the spatio-temporal analysis HTM nodes provide, which gives hope for accurate representations of speech and articulation probability distributions. Another motivating factor is that HTMs were modeled after the human neocortex, as mentioned before. Since speech perception is generally performed by parts of the brain within the neocortex [14], HTMs seem well-suited to the task of verifying the Motor Theory of Speech Perception, at least to some measurable extent.

3. RELATED WORK

3.1 Use of HTMs

Although the use of HTMs in speech processing is limited, there are a fair number of studies that are relevant to our project. A primary example that demonstrates proof of concept is the speech processing demonstration bundled with NuPIC, the Numenta Platform for Intelligent Computing [11]. In this demonstration, Numenta have trained an HTM to process speech and solve two problems: gender classification and speaker identification. The demonstration uses digital audio recordings of human speakers that undergo signal pre-processing. The experiment shows promising accuracy results for the inferring HTM [11]. As such, the experiments performed in this project using similar audio pre-processing are also likely to show promising accuracy results. However, Numenta has only demonstrated the feasibility of performing gender identification, whereas this project aims to produce HTMs that identify phonemes.

A Stanford study has also demonstrated the use of HTMs for spoken language identification [13]. Robinson, Leung, and Falco reported high utterance-by-utterance accuracies for three English and one French classifications. The HTM was able to reach near perfect classification between English and French languages with fewer than fifty training examples. In contrast, the proposed system will identify phonemes of the English language, aiming to improve speech processing technologies.

Finally, Doremalen and Boves have shown that using HTMs to recognize spoken digits holds promise [5]. This is an ideal result, as phoneme recognition is not conceptually far off from digit recognition, giving rise to more hope for promising results. The results of this study show error rates below 20%, despite the HTM system not being fully optimized for processing audio signals. This study also suggests that the present implementation of HTM learning algorithms may be suboptimal for processing certain types of inputs – specifically, those that encode information mainly in dynamic changes. This is because HTMs do not propagate learned input patterns as top-down feedback. The study only focuses on spoken digit recognition, however, while this project will aim to extend the specificity of recognition to that of phonemes.

3.2 Speech recognition

While there are many approaches to phonetic classification, not all employ a hierarchical configuration. One method making use of such a hierarchical approach employed a series of support vector machines to segment speech according to a hierarchical classification of phonemes [6]. This team constructed a decision tree model, where each node acted as a binary classifier. The main advantage of this approach was the elimination of multi-class support vector machines, a direct result of using multi-step classification. Unlike this system, the proposed HTM classification scheme will take full advantage of the hierarchical learning models provided by Hierarchical Temporal Memory.

Much work has been done to determine the best representation of speech signals for recognition tasks. One such study [9] evaluated the performance of five spectral-based and five cepstral-based representations. The details of these representations are defined in more detail below, however their use in this study suggests their usefulness in the proposed system. Using the TIMIT database, the team tested each representation on multiple classification models for various tasks including phonetic classification. Much of the performance difference depended on the choice of classifier. Because it is not known exactly how the proposed HTM method relates to previous attempts, this shows the necessity of trying a number of different representations to see which provides the best results.

Finally, studies have shown the benefits in general of incorporating articulatory data in continuous speech recognition. One such system, a hybrid Bayesian Network and HMM, achieved a higher accuracy while using both articulator and sound data than on either data source alone [8]. Their system also used the MFCC data representation, and the team demonstrated the usefulness of using the first and second derivatives of the cepstrum as a further aid in recognition. This provides an encouraging proof of concept of the utility of articulatory data in phoneme recognition.

4. SYSTEM MODEL

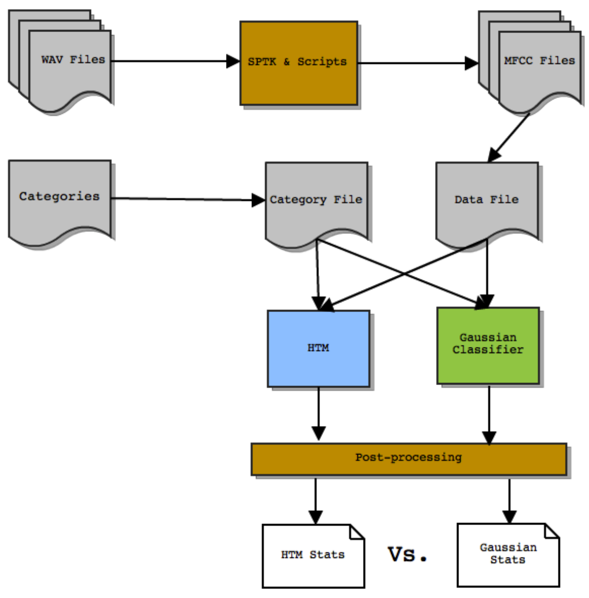

Figure 3: Model of Preliminary System

The goal of the experimental setup is to test the effectiveness of a given classification system. With this in mind, the model can be broken into two major components, a generalized scaffolding in which to run a given classifier, and a set of classifiers that are to be compared. The interface between these components is standardized, so that both reuse of code is high and experimental reconfiguration is agile. Figure 3 gives a graphical depiction of the model to help elaborate.

At a high level, the system prepares data for use by the classifier, runs some number of instances of the classifier, and performs basic post-analysis of the simulation. With these components in place, many different classifiers can be built to operate within the standardized interface, and their performance can be measured and compared. Thus, the experimental process involves designing a classifier, choosing some vector space of the parameters of the classifier, and plugging the classifier into the scaffolding to evaluate its performance.

Throughout the experiment, the Hillenbrand Vowel Database is used as the classification target. This set contains single-utterance recordings of approximately 200 speakers of varying ages and gender. Each recording in the database identifies a single vowel utterance, with the surrounding consonants the same across all recordings. 12 recordings of each of 12 vowels were recorded for each speaker. Samples are generally a few hundred milliseconds long, or approximately 10 thousand samples, in the WAV format. The amount of metadata in the database is rich, which allows for a wide variety of audio representations.

For each classifier, the data is partitioned into training and testing data. A number of audio transforms were compared in this experiment. One of these transforms is chosen and applied to the raw audio, and the entire data set is finally converted into a format usable by the classifiers. The experiment is then run by specifying the type of the classifier, and any run-time parameters are specified. Each classifier produces results in a standard format, so the results are tabulated by a standard suite of post-analysis tools.

Two major classification systems are evaluated in this experiment; Gaussian Classifiers and HTM Networks. A single Gaussian classifier was designed and tested for each unique data set to give a baseline performance metric. Then, a number of HTM configurations are configured. One variation between HTM networks is their network topology; variables include both the width and height of the hierarchy of spatial and temporal pairs. Another variation is the type of classifier node at the top of the network.

Finally, a number of numerical variables could be set for each configuration. This presents a large number of possible HTM configurations to test, and highlights the bulk of the experiment. Within a given HTM configuration, there is an amount of uncertainty present when choosing these values. The chosen solution in this situation was to iteratively run many classifiers across a wide vector space of parameters, and after post-analysis, determine a more targeted yet more fine-grain on which to run the classifiers again, with the final goal of finding the optimal parameter vector for a given HTM configuration.

5. SYSTEM IMPLEMENTATION

The model presented in figure 3 is implemented in four sequential major segments depicted in figure 4.

The first is the audio processing pipeline, which converts existing audio data from its original format into various formats applicable to statistical analysis. This corresponds to the top three elements of figure 3. Second, data is propagated through the HTM data pipeline, which converts, modifies, and formats appropriate files for use with HTM networks, the results of which are the "Category" and "Data" files observable in figure 3. Third is the machine learning experimentation, where learning and inference are preformed by the Gaussian and HTM networks. Finally, the results of the machine learning experimentation are post-processed to generate useful statistics and results, allowing one to measure the performance of the experiments.

5.1 Audio Processing Pipeline

Four major audio representations were compared in this experiment. Of the four, one was a hand measurement included in the Hillenbrand database, which was created during the original experiment associated with the recordings.

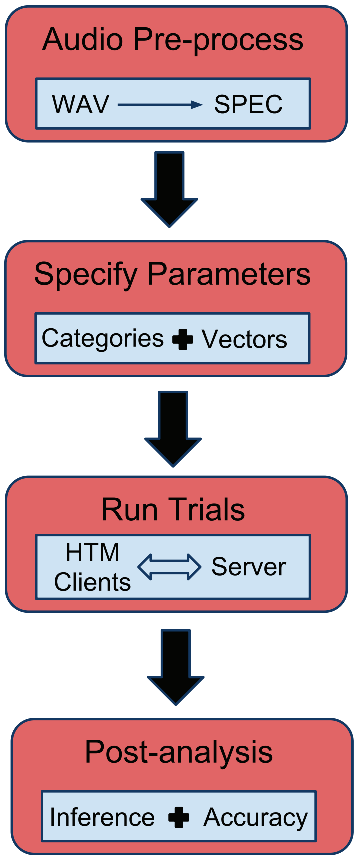

Figure 4: System Implementation

The remaining three were generated as a part of this experiment using different yet meaningful transforms. Again, the goal of using many audio transforms was to find a particular representation that worked best with the HTM networks.

The formants of a signal represent the dominant pitches that make up a sound. Formant measurements were taken from the Hillenbrand database directly. Each audio file had human annotated measurements for the first three formants of the speech signal, taken at ten percent time intervals throughout the vowel-part of the recording. The formant data was presented to classifiers in two ways. In the first case, a single vector containing all 27 measurements was used per recording. In the second case, nine separate three-dimensional vectors were presented as a time-varying sequence.

Three major audio transforms were compared in the experiment. These representations were generated by combining components of the Speech Processing Toolkit (SPTK) [3] with meta-data from the Hillenbrand dataset. A number of iterative transforms needed to be applied to the raw audio file. First, recordings needed to be truncated on both the leading and trailing ends of the file to exclude parts of the signal not associated with the vowel. The measurements were presented as time values, so these needed to be converted into sample counts. Audio was then framed using a standard 400 samples, overlapping at 80 samples, corresponding to a signal 25ms long with a period of 5ms. Finally, each frame was passed through a hamming window such that each frame was represented by a 512 dimensional vector, so that Fourier analysis could be performed. This pipeline so far was common to the remaining three audio transforms.

The first transform taken was a basic Fast Fourier Transform, using only the real-valued power spectrum measurements and discarding the complex-valued phase measurements. This transform produced a basic spectrogram without further scaling. The second transform was a log-Mel spaced spectrum. This involved using a FFT as well, but the frequency bands were then scaled log-linearly according to Mel function which approximates how humans recognize pitch. Finally, the third transform used was the MFCC, or Mel-frequency Cepstral Coefficients. These were obtained by performing a second FFT on the already calculated log-Mel spectrum. Each of the three generated audio representations were applied to each recording in the database, and the binary float-valued files for each were converted into a textual representation for Matlab and the HTM library.

5.2 HTM Data Pipeline

For any given instance of a classifier, both Gaussian and HTM, the set of all data is organized and partitioned into training and testing data. Since HTM networks impose strict requirements on data formats, it becomes necessary to involve custom tools and scripts that make the conversion more efficient as well as generalizable to different input formats.

The preliminary stage sorts all the available data into the relevant categories to which they will be mapped, and creates convenient subfolders and symbolic links that are later used as the input to HTM data generating scripts. In the current implementation using the Hillendbrand vowel dataset, the speaker files are sorted by specific gender, speaker, and vowel class.

Then, on a per-experiment basis, a sample subset of data is selected, for instance all male vowel speakers, and the list of files is permuted randomly and partitioned into training and testing data using a predefined ratio. Generally, a ratio of 4:1 train to test data is used. Finally, a generic HTM data generator is used together with learning parameters specific to the current experiment, to create suitable HTM training and testing data and category input files.

5.3 Gaussian Classifier

One relatively simple approach to machine learning is based on multivariate Gaussian distributions. A Gaussian distribution in one dimension is defined by a mean and a variance; in n dimensions, all data points are vectors of length n, and the unidimensional variance is replaced by an n-by-n covariance matrix. A Gaussian classifier is presented with vectors belonging to particular categories, and a distribution is constructed for each category by calculating the mean and covariance of the vectors belonging to it. To categorize an unlabeled test vector, the classifier determines which of its distributions is most likely to have produced that vector. The category corresponding to the distribution with the highest likelihood is the classifier's answer.
The audio transform used in this project, the MFCC, represents each 80-millisecond window of a file as a vector of 13 numbers. One option for presenting files to a Gaussian classifier is to concatenate all of these vectors together (adjusting as needed for variable file length) such that each file is represented as a single long vector. However, as these vectors grow in size, calculating the covariance matrix and

its inverse becomes increasingly susceptible to error from floating-point imprecision. Our current implementation uses only 4 samples from each file, out of around 100 MFCC vectors on average. One way to use all possible data would be switching from Matlab to Python (with NumPy) to use the decimal object - Matlab does not have a data type capable of arbitrary-precision. Alternatively, the samples taken from the files could be augmented with delta and double-delta information (that is, first and second derivatives) and MFCC vectors could be presented for classification one at a time. That is, classification could be performed on each 80-millisecond window instead of each file - this is similar to the HTM's procedure. These avenues, and others, will be explored in the future.

5.4 HTM Classifier

HTM networks can be designed to accommodate any number of input nodes and levels, but these should correlate with appropriate models of learning in order to infer accurate results. In this experiment, the HTM designs incorporate both the biological learning models for human audition as well as the structure of the audio data being processed.

5.4.1 Basic HTM Classifier

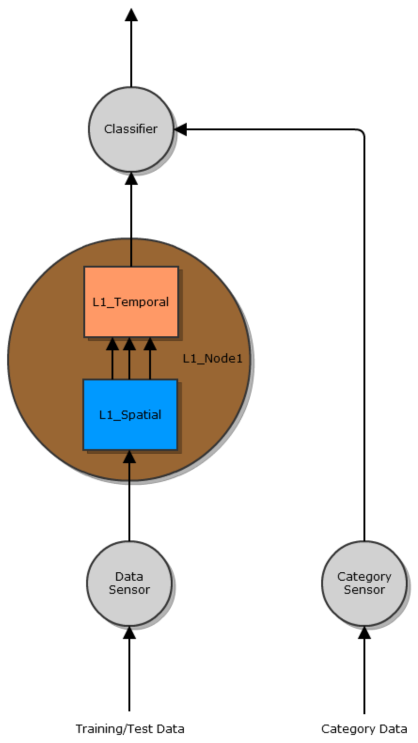

Figure 5: First Prototype, A Single Node HTM

The design in figure 5 represents a prototype HTM used as a base model for testing. It simply reads in all data inputs, and trains the HTM with only one node. This model was used as a testing site for new data formats, learning parameters, and other subtle changes that might need a substantial amount of debugging, before exploring the results of such changes on more complicated HTM designs that take substantially longer to run.

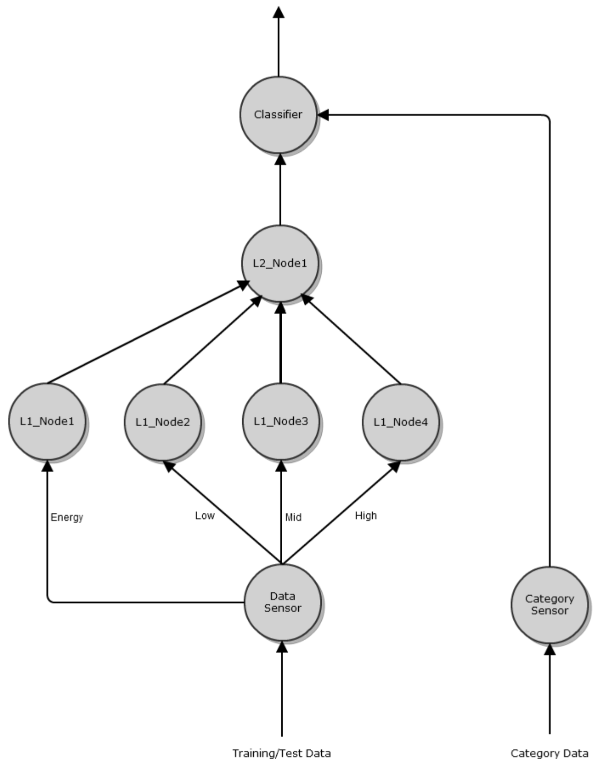

Figure 6: Modeling with Energy

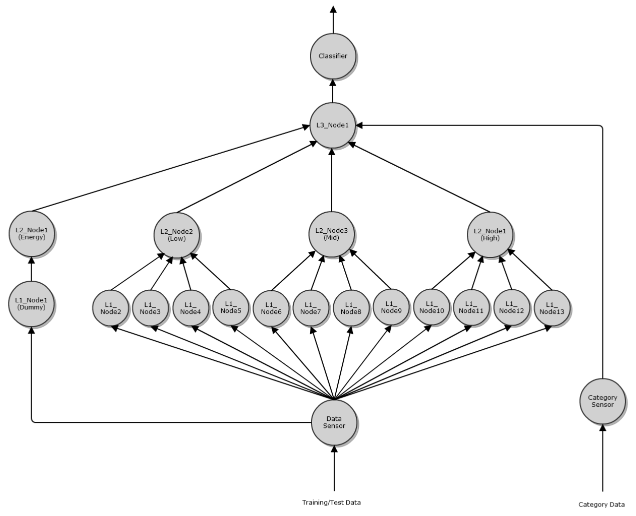

Figure 7: Fully Distributing the Data

5.4.2 Variations of the HTM Classifier

Configuring an HTM network involves a number of design choices. Variations on the basic HTM classifier include changes to the network topology, the classification strategy,

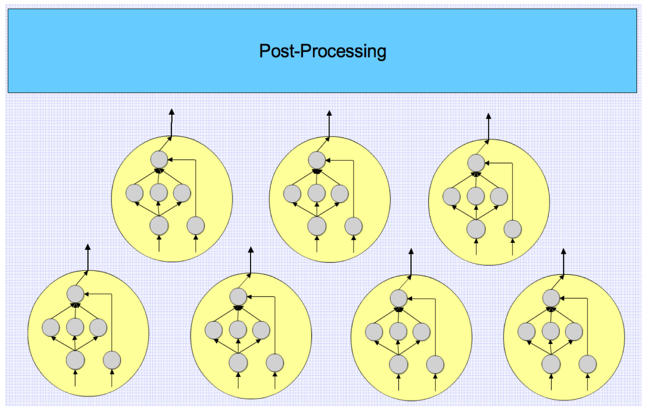

Figure 8: Binary Classification of Vowels or Phonemes

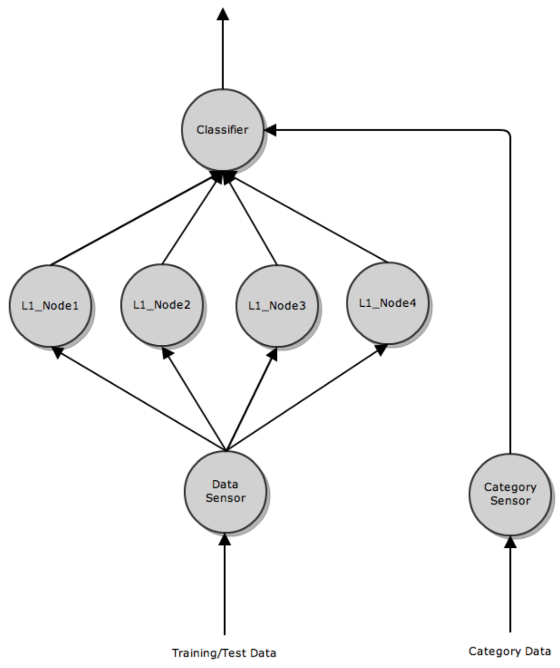

Figure 9: HTM with Four Nodes

and a range numerical arguments to the training process. The large number of possible configuration due to these degrees of freedom required a large number of tests, which was addressed by developing the parallel experiment harness, discussed in the next section.

A number of different network designs were attempted. The two main variables involving network topology are the width and depth of the hierarchy of spatial and temporal node pairs. As noted in the previous subsection, the basic HTM classifier uses a single level with a single node. When testing using MFCC data, networks were designed that took advantage of the features of the data, such as the design depicted in figure 9. MFCC vectors carry information about 12 frequency bands and 1 energy band. One network was designed that split the energy separate from the frequencies, and split the frequency sub-vector into high, medium, and low slices. Another network was designed, but not implemented, that gave each frequency band its own pooler pair at the lowest level, with a 12-3-1 fanning hierarchy, as seen in figure 7. Other variations of the network topology were designed and implemented with the FFT and Log-spectrum data. The data vectors of these audio transforms always had dimensions that were powers of two, so networks with 4 pooler pairs, with one (figure 9) or two (figure 10)levels, were also implemented and tested.

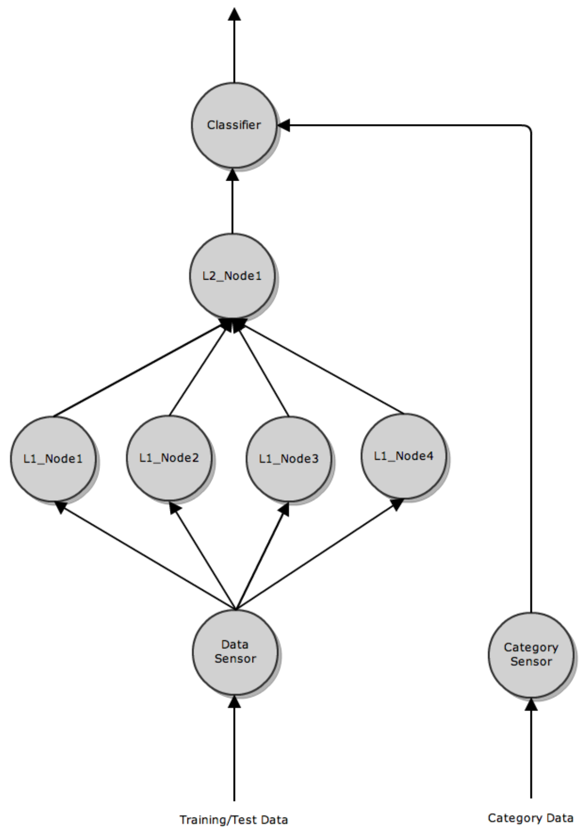

Figure 10: HTM with Four Nodes: 2nd Level Inference

The classification node at the top level of each HTM network could be either a Naive-Bayes classifier or a Support Vector Machine. Both types of classification nodes were used on each network designed to compare their relative performance. Support vector machines were also used with special data sets with a binary classification goal; rather than attempting to identify each vector by its vowel category, a single vowel was chosen out of the twelve, and the HTM attempted to decide whether each vector was or was not an instance of that particular vowel 8.

Finally, there were three major numerical arguments to the HTM training process itself that were varied over each configuration instance. First, the requested coincidences of the spatial pooler nodes gave a maximum bound on how many coincidences could be recorded as a result of new, dissimilar vectors in the training set. Second, the requested groups of the temporal pooler nodes gave a maximum bound on how many groups temporally local coincidences could be recorded over the course of training. Lastly, the max distance parameter was the threshold determining if a new vector should be recorded as a member of an existing coincidence, or if a new coincidence should be created as the new vector was sufficiently far away.

All of these variables were combined to produce a large number of configurations and their associated vector spaces of parameters. After each classifier was implemented, a permutation of the data set was generated, and the classifier was run from within the test harness. ItÕs results were recorded in a folder named by itÕs parameters, and the post-analysis suite was used to generate a summary of the results for that instance.

5.5 Experimentation and Parameter Testing

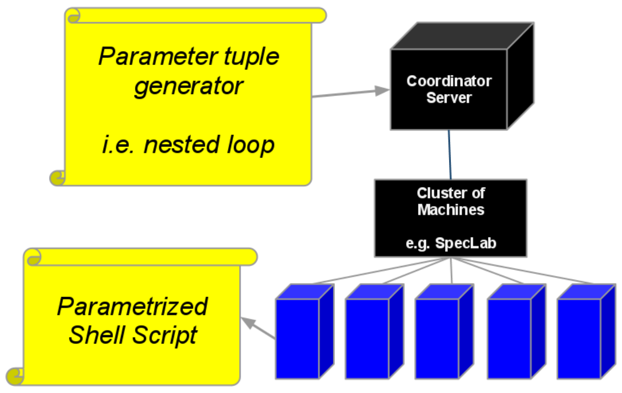

Figure 11: The HTM Parameter Testing Harness

During the later stages of the experiment process, it was observed that more complex simulations could require many hours of machine time before producing usable data. As a result, an experiment harness was created to automate the process of running many instances of simulations with varying vectors of parameters in parallel across multiple machines. Two programs, a coordinator and a drone, were written in Go to facilitate this automation and speed up the turnover between the planning and analytics phases in our experiments. A list of vectors would be generated programmatically using nested loops and piped to an RPC server in the coordinator. Drones would make synchronous RPC calls to the coordinator to get a single vector, and would fork a child process to run the simulation with the specific arguments.

This experiment harness was deployed using a mixture of personal and shared resources. After the coordinator was deployed on a personal computer, multiple drones would be spawned across the SPEC Lab machines. Since the running time of an experiment was much longer than the transaction time to fetch a new vector, this configuration allowed for a speedup by a factor of 10-20, depending on the load of the shared servers. Figure 11 shows a graphical depiction of the harness system.

6. SYSTEM PERFORMANCE

In both classification systems, a confusion matrix is used to describe the accuracy and performance of the model. For a task involving k classes, a $k \times k$ matrix is generated after training and testing. Each element of the matrix represents

Table 1: Confusion matrix for Gaussian Classifier

19	1	0	2	0	1	1	0	0	1	0	0
1	19	5	3	0	1	0	0	0	1	1	0
0	7	19	1	0	1	0	0	1	0	3	0
2	0	0	17	0	0	1	0	0	2	3	1
0	0	0	0	18	0	5	3	0	0	0	2
0	0	0	0	0	19	0	0	0	1	0	2
1	0	0	3	2	0	17	0	0	1	0	2
0	0	0	0	1	0	3	25	0	0	0	0
0	0	0	0	0	0	0	0	23	0	1	8
0	0	1	0	0	0	0	0	0	22	1	1
0	1	3	0	0	0	0	0	1	13	13	0
0	0	0	0	1	1	0	0	2	0	0	23

Table 2: Confusion Matrix for Single Node HTM Classifier

0	0	0	0	0	0	0	0	0	96	0	0	0
342	108	0	0	0	0	0	0	0	1	0	0	0
344	0	107	0	0	0	0	0	0	0	0	0	0
360	0	0	118	0	0	0	0	0	0	0	0	0
257	0	0	0	81	0	0	0	0	0	0	0	0
338	0	0	0	0	110	0	0	0	0	0	0	0
348	0	0	0	0	0	118	0	0	0	0	0	0
263	0	0	0	0	0	0	64	0	0	0	0	0
365	0	0	0	0	0	0	0	65	0	0	0	0
406	0	0	0	0	0	0	0	0	71	0	0	0
277	0	0	0	0	0	0	0	0	0	48	0	0
277	0	0	0	0	0	0	0	0	0	0	55	0
372	0	0	0	0	0	0	0	0	0	0	0	53

a count, where the value at position (i, j) represents the number of times a vector generated by the i-th class was recognized as an element of the j-th class. In both cases, the classifier prints for each test vector the likelihood that the vector was generated by each class, and the index with the highest such probability is interpreted as the classification. An accurate model will produce a perfectly diagonal matrix, while a non-diagonal result matrix suggests issues with the model or its implementation.

6.1 Gaussian Performance

The Gaussian classifier's accuracy was quite good, as Table 1 shows. Most of the responses were correct, with a few random errors and one systematic one (confusing vowel 10 with 11, mistaking many vowels for vowel 12, etc.). Further enhancements to the Gaussian implementation revealed even better results, running on both formant data and Mel Cepstrum audio transforms. As such, a proper baseline for comparing HTM performance has been achieved.

6.2 HTM Performance

The HTM networks throughout the experiment performed poorly on the vowel classification task. During the long-term experiment, where multiple configurations were tested, it became clear that the variable that had the most effect on the performance of HTMs was the distance threshold. For instances with too small of a distance threshold, all training vectors would be mapped to distinct coincidences in the spatial pooler nodes. This is undesirable as collisions in the spatial pooler are necessary to recognise common temporal patterns. On the other extreme, for instances with too large of a distance threshold, all training vectors would be mapped exclusively to the same coincidence in the spatial pooler node, and that lack of variance would present the same problem in the temporal pooler.

This pattern was confirmed early in the experiment, and led to the method of trying to find the best compromise for the distance metric, which was facilitated by the parallel test harness. Unfortunately, despite multiple iterations of narrowing the search range on different HTM configurations, the region between extreme low and high distance thresholds did not contain a maximum reachable by the optimization strategy. Furthermore, many configurations would produce an identical confusion matrix for every trial, where every vector in an entire data set would be mapped to the same category for no particular reason. No information could gathered from these instances, and experiments involving their respective configurations were frozen.

7. FUTURE WORK

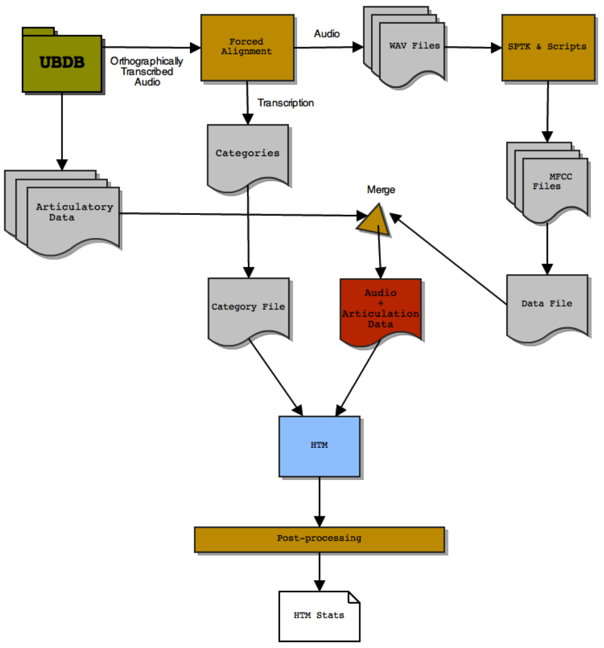

Figure 12: Model of System Incorporating Articulatory Data

What are now considered future directions for this project were intended in prior planning to be part of it. This is true of effectively implementing HTMs, expanding identification from isolated vowels to continuous speech, and employing articulator trajectories alongside audio data.

For the reasons outlined above, HTM networks hold substantial promise for speech processing tasks. The enormous scope of HTM implementations (with many possible values for each node's parameters, plus a wide range of options for connecting multiple nodes) allows hope that further investigation will yield more successful systems.

Processing continuous speech is a more applicable problem than vowel identification, but it is much more difficult. Auditory effects not present in the vowels used in this project cause serious headaches for modern speech recognition, but there is reason to think that these would be mitigated at least in part by the use of HTMs due to their hierarchical pattern-matching structure.

Finally, using articulatory X-ray data to aid in phoneme recognition was another potential component of this project. This would constitute a formal test of the Motor Theory of Speech Perception, which to date remains in the theoretical domain, and a successful result would have major implications for cognitive science.

8. CONCLUSIONS

A possible result of this experiment is that HTM classifiers are not well suited for speech recognition tasks. However, other experiments confirm, at least to some degree, that HTM networks can perform basic recognition tasks involving human speech. [13] [11] Furthermore, properties of human speech and the biological mechanisms of speech perception suggest that HTM networks are a good fit. Not only is speech time varying and often periodic, but the way it is presented to the brain as a spatial signal of neural activity would seem to make the audio representations employed in this experiment a good fit for the HTM algorithms.

Another possible explanation to the consistently poor results, is a possible incorrectness in audio transforms. This possibility is quickly ruled out, however, by the success of the Gaussian classifier described above on the audio transforms performed in this paper. The possibility of an incorrect, corrupt, or even out of date NuPIC library also comes to mind. However, because the provided NuPIC samples function correctly and provide consistent results, it cannot be concluded that any inherent defect in the NuPIC library exists.

While the initial response to these results is that the HTMs are not being implemented correctly, any conclusive evidence to support an incorrect implementation of the HTM networks is amiss, due to malfunctioning NuPIC analysis tools. Furthermore, lack of official support from Numenta, as was previously available prior to December 2010, made debugging information scarce. As a result, HTM debugging has been painfully slow and unproductive, dealing mainly with technical issues surrounding the NuPIC library, instead of focusing on potential improvements to design, input parameters, or data format as was originally planned.

On the other hand, despite having closely followed the NuPIC documentation, inconsistencies between the documentation and provided NuPIC examples have made it difficult to ensure a proper implementation altogether. In an effort to assert correctness of the HTM networks, the source code for each network adhered to certain "best practices" that were identifiable in the provided examples, since they represent testable functioning HTM networks.

Future releases of Numenta's learning algorithms might supply a new development platform for HTM networks, however, it will certainly reopen the official support channel they had previously shut down for the duration of this project. With this in mind, we look forward to pursuing our future work in this field, to be able to conclusively judge the effectiveness of HTMs in processing human speech.

8.1 Ethics

The work presented in this paper poses no threat to the safety, security, or privacy of any potential users or researches. While some scientific experiments have clear ethical issues surrounding them, the implications of HTM use in speech recognition are the same as any other technology that is currently being used. It is simply a different statistical model, one that is inspired by biological theories, that could have possible positive effects on the performance and accuracy of current technologies.

We have investigated the potential for HTM technology to be misused in more general settings, and can conclude that in essence, these are the same potential risks other machine learning models might pose. To discuss the implications of artificial intelligence and the ethical debates that follow, however, is a large deviation from the main focus of this research.

9. REFERENCES

[1] J.S. Garofolo, L.F. Lamel, W.M. Fisher, J.G. Fiscus, D.S. Pallett, and N.L. Dahlgren. *Darpa Timit: Acoustic-phonetic Continuous Speech Corps CD-ROM*. US Dept. of Commerce, National Institute of Standards and Technology, 1993.

[2] Dileep George and Bobby Jaros. *The HTM Learning Algorithms*. Numenta Inc., March 2007.

[3] SPTK Working Group. Speech signal processing toolkit (sptk). 2010.

[4] Jeff Hawkins and Dileep George. *Hierarchical Temporal Memory; Concepts, Theory, and Terminology*. Numenta Inc., March 2006.

[5] Interspeech. *Spoken Digit Recognition using a Hierarchical Temporal Memory*, September 2008.

[6] A. Juneja and C. Espy-Wilson. Speech segmentation using probabilistic phonetic feature hierarchy and support vector machines. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 1, pages 675–679, 2003.

[7] A.M. Liberman and I.G. Mattingly. The motor theory of speech perception revised* 1. *Cognition*, 21(1):1–36, 1985.

[8] Konstantin Markov, Jianwu Dang, and Satoshi Nakamura. Integration of articulatory and spectrum features based on the hybrid hmm/bn modeling framework. *Speech Communication*, 48(2):161 – 175, 2006.

[9] H. M. Meng and V. W. Zue. Signal representation comparison for phonetic classification. In *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*, pages 285–288, 2002.

[10] Numenta Inc. *Hierarchical Temporal Memory; Comparison with Existing Models*, March 2007.

[11] Numenta Inc. *Speech Processing with Hierarchical Temporal Memory*, June 2008.

[12] F. Pulvermüller, M. Huss, F. Kherif, F. Moscoso del Prado Martin, O. Hauk, and Y. Shtyrov. Motor cortex maps articulatory features of speech sounds. National Acad Sciences, 2006.

[13] Dan Robinson, Kevin Leung, and Xavier Falco. Spoken language identification with hierarchical temporal memories. December 2009.

[14] Robert J. Zatorre, Marc Bouffard, and Pascal Belin. Sensitivity to auditory object features in human temporal neocortex. *J. Neurosci.*, 24(14):3637–3642, 2004.