Spoken Language Identification With Hierarchical Temporal Memories

Dan Robinson
danjr89@stanford.edu

Kevin Leung
kkleung@stanford.edu

Xavier Falco
xfalco@stanford.edu

December 11, 2009

1 Introduction

The task of spoken language identification has enjoyed many promising approaches via machine learning and signal processing since the late 1970s. However, most either employ Hidden Markov Models (HMMs) to model sequential data or use language-dependent phoneme recognizers as primary features [3, p. 33]. The former has sparsity-related issues and is sensitive to previously unseen events, which are common in human language. The latter requires extensive labeling of training data on a phonological or prosodic level, which is often impractical, and depends upon specialized information for each spoken language under consideration.

Our project is an attempt to use Hierarchical Temporal Memory (HTM) to do spoken language identification. HTM, implemented and distributed by the Numenta company, is a new technology inspired by the human neocortex. Distinguishing characteristics of HTMs include their ability to natively process both temporal and spatial information and their potential for deep hierarchical structure. These two features of HTMs suggest their utility in audio and language tasks.

2 Hierarchical Temporal Memory

An HTM is a tree-shaped hierarchy of nodes, where each node implements a learning and memory function and thus encapsulates a primitive algorithm. Nodes are organized into *layers* or *levels* and don't interact directly with other nodes of the same layer. Lower level nodes receive sensory data and output *coincidence* predictions to higher levels. In this way, an HTM network abstracts information as it is passed up the hierarchy.

During training, the HTM is presented with examples of data as it changes over time. The temporal element is critical; the algorithm expects input that changes over time in predictable patterns, and it attempts to discover and learn these patterns. This makes HTMs particularly suitable for audio processing. Previous research on audio classification using HTMs includes promising results in the training of HTMs to recognize spoken digits [2].

Each node in an HTM is made up of a *spatial pooler* and a *temporal pooler*. The former quantizes and clusters input data to the node, and the latter learns temporal patterns amongst incidences of these quantization points. (For simplicity, one can conceive of the former as a preprocessing step for the latter, which substantially reduces sparsity.) There are many specific algorithms for spatial and temporal pooling, several of which are implemented in the Numenta release; for more information, see `http://www.numenta.com/for-developers/software/prog_guide/Output/prog_guide-05-1.html`. We used the standard NuPIC Gaussian spatial pooler and sumProp temporal pooler throughout our code.

3 Dataset Description

Thus far, our dataset includes approximately 8400 utterances from each of American English, British English, French, Russian, and Japanese. (We have and would like to incorporate data from 21 other languages, but have been unable to do so thus far due to AFS space constraints.) Most utterances range from one word to one short sentence and are less than 4 seconds long. Utterances are spaced evenly over a variety of speakers, which represent many ages and genders.

4 Experiments

So far, we have run experiments on the following 3 language classification tasks, with varying amounts of training data:

1. American English vs. French
2. American English vs. British English
3. Four-way classification between American English, French, Japanese, and Russian

5 Algorithm Details

As input to our HTM, we used a log-linear Mel spectrogram of our data files, taken with 64 frequency bins at 512-frame increments over our audio. As our data files have a sampling rate of 8kHz, this corresponds to 64 ms-wide frames. Most speech recognition literature seems to use some variant of this cepstral feature model [2][1], which also has some validity as a biologically inspired model of the mammalian cochlea.

When using HTMs for classification, a single classifier node is added to the top of the HTM, which produces a frame-by-frame likelihood distribution over the relevant classes (in this case, our 2-4 languages). We experimented with off-the-shelf KNN and SVM models. An HTM application has many other configurable variables, including the number of layers, number of nodes per layer, and a handful of variables internal to the nodes at each level. We refer to the former two of these as the *HTM architecture*, and the latter as the *internal node variables.*

The frame-by-frame classification results can be converted into classifications for our utterances in several ways, such as taking the average, product, or max over the frame-by-frame classification likelihoods to chose a most likely class. Intuitively, even 2-second long utterances have over 16000-frame samples. With frame-by-frame likelihoods that are even slightly better than random guessing, taking products or averages of distributions can produce a strong utterance classifier.

6 SVM Results

Our best utterance-by-utterance accuracies achieved were 98.75% for English vs French classification, 92.5% for American Enlgish vs British English, and 92% accuracy for the four-language classification task. All three of these were achieved with one layer of four nodes, using SVMs as classifiers. All three of these used at most 100 utterances per classifier, with the latter two using at most 50. Note that, in the latter test, our frame-by-frame accuracy was only 39.08%, demonstrating the powerful multiplicative effect of a weak classifier taken over many thousands of frames.

Thus, a one level HTM with an SVM classifier is able to reach near perfect classification between English and French with fewer than 50 training examples. This architecture's performance on the other tests is less impressive but still of significantly higher accuracy than any architecture using a KNN classifier.

Confusion tables from our multiclass classificatons of highest accuracy are included below for each classifier type. (Both use this one layer, four node HTM architecture.)

	Eng	Fre	Jap	Rus
Eng	25	15	0	0
Fre	0	10	0	0
Jap	0	0	24	0
Rus	0	0	1	25

	Eng	Fre	Jap	Rus
Eng	20	2	0	0
Fre	0	23	0	0
Jap	5	0	24	0
Rus	0	0	1	25

Table 1: Best utterance-by-utterance confusion tables with frame-product classification, using KNN (left) and SVM (right) classifiers, trained on 550 and 30 utterances per language, respectively.

Unfortunately, the computation time required by our SVM classifier on larger training sets was prohibitively long. What's more, though this SVM architecture does successfully identify spoken languages, we wanted to investigate different HTM architectures, so as to empirically compare the quality of the discovered features at each layer. With these two facts in mind, we resolved to pursue other architectures and node configurations, using only KNN classifiers.

7 Varying The HTM Architecture

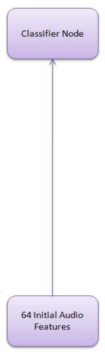
For the remainder of our project, we attempted to modify our HTM architectures and internal node configurations to exploit the potential of HTMs to discover deeper audio features and, hopefully, more accurately classify our languages.

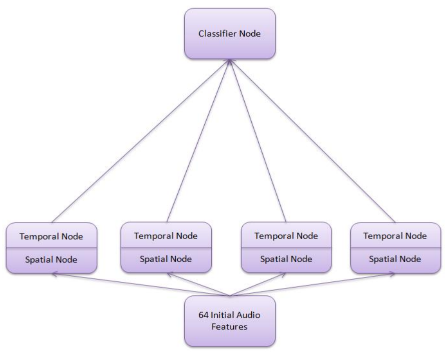
We found the default internal node values performed the best for all five of the parameters we considered. Modifying any of them independently decreased our classification accuracy at both the frame and utterance level. Though these parameters interact in deep and complex ways, our primary goal was to explore various HTM architectures, not to hand tune internal nodes. We settled on the initial parameters and held them constant throughout our remaining tests.

In addition to a baseline architecture, in which initial input features are fed directly into our frame-by-frame classifier, we explored three single layer HTM architectures, two configurations with three layers, and one of two layers.

Our one layer structures used two, four, and eight nodes, splitting the 64 initial input features evenly. Of these, the four node structure performed uniformly better, and it is the only single layer HTM we consider henceforth.

Our first three layer structure consisted of four nodes at the first layer, two at the second layer, and a single node at the top layer, using the output from the top layer as input for our classifier node. Our second three layer structure had a similar layout, but with the concatenated output from every layer as input for our classifier. (That is, at every frame, we passed the output from all seven nodes directly to our classifier.) Diagrams of the four HTM architectures we consider further are presented below.





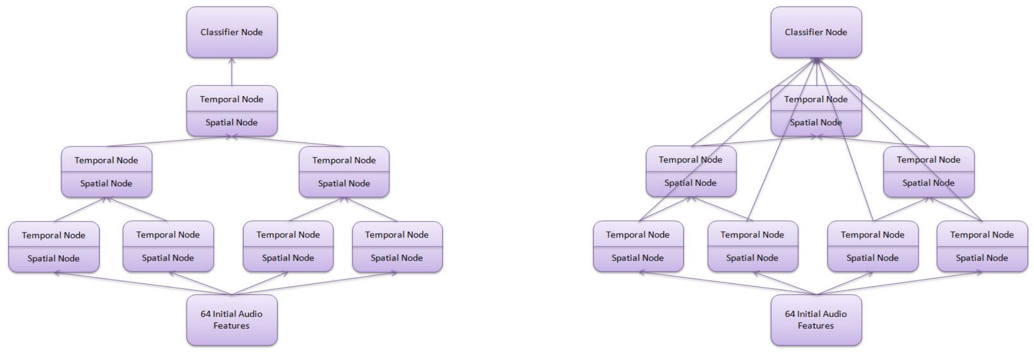


Table 2: From left to right, top to bottom: our baseline classifier, single layer HTM with four nodes, three layer HTM using only top layer output, and our three layer HTM using all three layers of output.

8 Binary Classification Results And Conclusions

Between our binary classification tasks, relative performance between our HTM architectures remained constant. Performance was generally better when classifying between British and American English than between English and French. Learning curves for these two tasks are presented below, with our SVM results added for perspective.

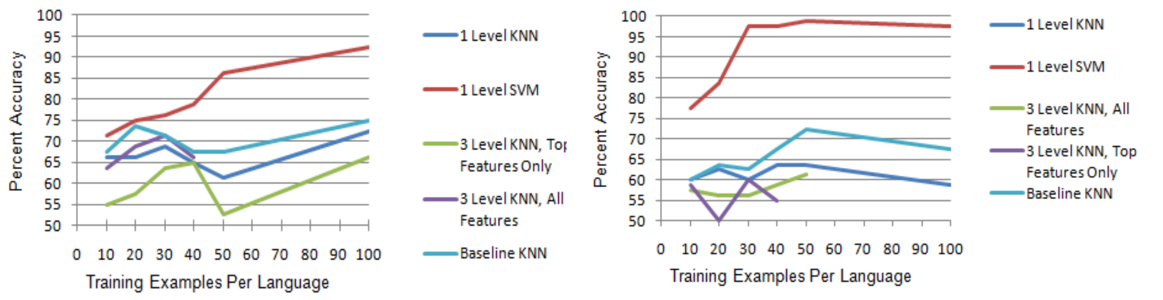


Table 3: Training curves for our two binary classificaton tasks, with British English vs American English on the left and French vs American English on the right

Amongst the configurations with a KNN classifier, our baseline system performed best, followed by the one level architecture, the architecture using features from all three layers, and lastly by that using only our top level features.

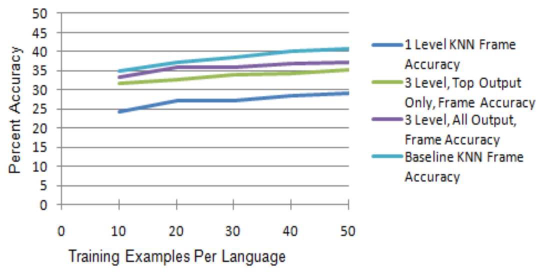
It seems that, for these tasks, additional HTM use decreases performance. Indeed, adding a first HTM layer from our baseline causes a uniform decrease in accuracy accross the training curve, as do adding more layers on top of that. Even when adding the second and third layer outputs as features to a model using just the first layer outputs, we hurt our overall accuracy.

There are several possible analyses of this phenomenon. First, the training of the HTM layers is completely unsupervised; the patterns it learns at each level need not have any correlation to our tasks. It may well be that the unsupervised portion of our learning produces a feature set that is less and less useful for our task at each layer. Indeed, fundamental to the training is clustering of input, which may obscure some of the intricate, low-level information necessary for language classification.

Additionally, note that the variance of our model increases significantly with each level of input added. Indeed, the first layer typically learns on the order of 500 features, made up of spatiotemporal patterns from the original 64. The total number of features from our three-layer model is often greater than 20,000. With this forty-fold increase in the number of features comes a large increase in variance, compensating for which with equally massive training sets would be impractical given reasonable limits to our computational resources. Noting that our training error is usually 0% at the utterance level, this variance issue may well be our chief source of error. These deep HTM networks may merit further work, either with more supervision in the training of each level or with significant pruning of feature sets, which we can accomplish with standard model selection techniques.

9 Four Language Classification Results And Conclusions

The results from our four language classification task differ in that our one level HTM configuration does outperform our baseline system. What's more, this system is uniformly worst in accuracy at the frame level, and yet is best in utterance classification accuracy. Learning curves for frame and classification accuracy are presented below.



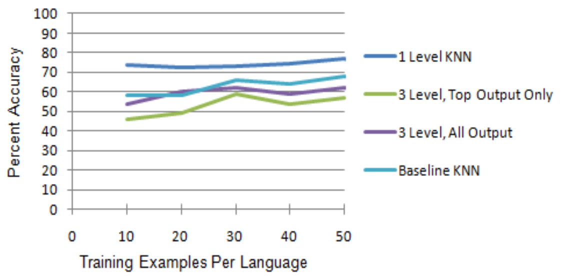


Table 4: Training curves for our four language classificaton task, frame classification accuracy on the left and utterance classification on the right

This discrepancy between frame accuracy and utterance accuracy can be explained by the fact that our one level HTM yields the most balanced set of frame errors. The baseline KNN system produces a frame by frame confusion table that is highly lopsided. (That is, between any two languages, well above half of the frame classifications align with one of the two languages.) This phenomenon yields a high error rate, whereas the one layer HTM produces a frame confusion table with a large numbers of errors, but which is nearly symmetrical A system such as the latter is more favorable for use in utterance classification, in which individual frame misclassifications matter very little but statistics over thousands of frames are critical.

Thus, for language classification involving more than two languages, it does seem that HTMs add some value, as our best performing system did utilize our lower level HTM features. With further tuning and improved feature filtering, one may be able to classify spoken language in close to real time, and with high accuracy. That said, current HTM implementation does not seem suited for binary language classification.

References

[1] H. Li, B. Ma, and C.-H. Lee. A vector space modeling approach to spoken language identification. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(1):271–284, 2007.

[2] J. van Doremalen and L. Boves. Spoken digit recognition using a hierarchical temporal memory. *Proceedings of the 9th International Conference of Interspeech*, pages 2566–2569, 2008.

[3] M. A. Zissman. Comparison of four approaches to automatic language identification of telephone speech. *IEEE Transactions on Speech and Audio Processing*, 4(1):31–43, 1996.