MOBILE PHONE BASED INFERENCE MODELS USING PEOPLE-CENTRIC FEATURES CS 134: FINAL REPORT

NICHOLAS D. LANE

1. Introduction

Robust activity recognition, the classification of what a person is doing based on low level sensor data, is a difficult learning problem. Mobile phones are an ideal platform to collect sensor data about people, given their ubiquity and increasing ability to do sensing, computation and storage - nevertheless doing effective activity recognition is still a hard problem. In this project we are interested in leveraging the ability of people to distill information from their surroundings and themselves to make this problem simpler. Here we are looking at a single specific example of this idea, namely - the use of spoken words (captured using from audio sampled by a mobile phone and then to which speech recognition techniques are applied) as features to activity classification. We hope that by using words as features we can model the activities as simple collections of these words providing a simple and effective structure.

1.1. **Summary of Accomplishments.** The accomplishments of this project are as follows:

(1) Collected audio data from a number of activities, transcribed sections of this audio, and tested the use of simple document classification techniques to recognize activities.
(2) Built a simple speech recognizer that runs on the iPhone.

2. Experiment

We want to explore the idea that spoken words captured and recognized when people are performing activities would provide effective features to perform forms of activity recognition. Our hypothesis was as follows: *Only a small fraction of ambient spoken words are required to be recognized during an activity - for these to support activity recognition. These words would act as robust features able to perform complex types of activity recognition while only being used in a simple model, such as a bag-of-words model that otherwise maybe used for document classification.*

2.1. **Methodology.** To evaluate this idea we perform a two prong feasibility study.

Firstly, we build a prototype word recognizer that is able execute on a mobile phone, to demonstrate that this aspect of the proposal is feasible. We perform some simple evaluation of its performance but do not evaluate it at any depth since it is far below the state of the art and was built for educational purposes.

Secondly, we investigate directly the suitability of spoken words to support activity recognition by removing the problem of speech recognition from the process entirely. Instead we loosely transcribe fragments of collected audio and use this as input data to 'bag-of-word' models that attempt to infer an activity. We collect in total around 19 hours of audio (this figure is actually lower than the milestone figure because even though we collected a little more data, quite a lot of data was ignored, as the set of classes to be recognized was tightened) while performing a range of activities. These activities being: *going to the bank, going to pizza, going to get a coffee, going to the gym, going to the library, going to a bar, going to the gas station, going to the bookstore, going to burrito place.* There were an roughly even about examples in the dataset of these activities being performed. Approximately 10 to 12 times for each activity, with some large exceptions being the number of times going to the coffee shop was captured. These were not the final

	sample 1	sample 2	sample 3	sample 4
word accuracy	11%	9%	18%	31%

Table 1. Estimated accuracy rate of coarse transcriptions that were performed.

classes we attempted to classify however, just the underlying activities that were done as part of data set collection. The audio data was captured from an Apple iPhone that was either in a pocket or carried in the hand. We have an existing application that was developed in the lab for data collection, wave files were collected.

Our transcription process was done only by sampling fractions of the audio traces. We transcribed in a best effort fashion 5 to 10 minute chunks of all individual activities performed. The audio was listened to and any words that could be made out were written down, however if any words were missed they were not replayed to be heard again. At times during the transcribing process the words came to quickly, or even more commonly there were many words said in the background that could not be clearly heard.

To get an estimate of how many words were being captured 4 different 5 minute segments that were transcribed were picked out. These were listened to again but this time they were listen to just only count what was believed to be words said. Many background words were overhead but could not be made out. This number was used to estimate how many words from the total that could be potentially heard, were actually transcribed. We were interested in this number to since in a real setting the actual percentage of words that could be recognized would be low. So we need to be sure our word analysis was not based upon a percentage that was unreasonably high. Part of the point to keeping transcriptions course was to keep this figure low (and to keep the transcription process as low overhead as possible). These figures were surprisingly low but it did make sense given he large number of background words that could almost be heard but not really, in a number of the samples that were used there was a lot of background conversations going on

3. Implementation

3.1. **Mobile Phone based Word Recognizer.**

A simple isolated word recognizer was written that runs directly on the iPhone. It is a proof of concept implementation that demonstrates the potential for light-weight word recognition to be run directly on the hardware of devices such as mobile phones. This software was written in C++ and can both train models and evaluate models directly on the phone (see the source code in `prototype` directory of the code submission).

The recognizer process operates as follows: audio signals (in wave format) are divided into non-overlapping frames of 30 ms. 12 MFCC coefficients are calculated for each of these frames. These MFCC feature vectors are quantized based on a 512 entry code-book. A code book is built at training time and feature vectors are assigned to the code words based on euclidian distance. Every word that is recognized is modeled by a 10 state discrete HMM that uses the code word symbols as possible observations. We use the standard forward algorithm for HMMs to determine the likelihood of each mode based on a sequence of observed code words. The word model selected as being the inferred word based on the audio is the one with the highest probability.

The training of the system is done by grouping labeled examples audio segments containing only a single word. Each of these groupings are used to train individual HMMs. The forwards-backwards algorithm is used to train these HMMs. The code book that is used is built using the LGB algorithm, and all the training data. This algorithm produces the best series of code words (constrained by the dictionary size limit) to represent the features it is presented with.

The recognizer is implemented as a series of independent programs, applying the UNIX philosophy of single programs doing one thing and doing that thing well. These programs interact by piping the out of one as the input of another. Decomposing the operations in the recognition process into single programs makes them more easily combined to run experiments to test parameters or doing accuracy measurements.

Support scripts for training and performing inference are found in the `prototype_eval` directory of the code submission.

In the implementation code is used from a number of sources as well as being written by myself. The LBG algorithm (Linde, Buzo, and Gray) that generates the code book was sourced from [3]. The MFCC library was written by Cory Cornelius on a prior project he did on speaker recognition. The HMM implementation uses functions that were sourced from [4] and were modified in some places to make small fixes that were necessary to make it work under C++ running on the phone. The design of the recognizer code, how it operates, how it uses the other functions and all other auxiliary code (e.g., pipelines to train and evaluate models) was written by myself. The only part of the training process which is not done on the phone is the construction of the code book, which is a MATLAB script. This could have been easily ported to C++ which I plan to do in future versions to make the implementation self contained and allow the phone be able to learn new word models by itself. The phone is able to recognize words entirely independently.

3.1.1. *Limitations.* The implementation does not perform continuous recognition of words based on a live stream of sound. Instead wave files are captured over time and then recognition can be done on the phone as a batch operation based on the audio collected. Another functional draw back is the lack of segmentation routines for audio data. Currently segmentation of audio data is assumed - and not performed by the system. In our evaluation the audio fragments that are used in the recognition process only contain a single word - so segmentation is not necessary.

Another large limitation is the lack of support to handle noise. All training was done with data from a clean environment which will have a feature profile which is different to audio from a noisy environment. We could train based on more nosey settings - or adopt solutions such as removing the mean from both the training and testing data to normalize some of the differences between the test and training sets.

A large limitation is the fact word recognizer has only been trained and tested on one person. It would definitely fail to recognize words in the diverse set of contexts phones that this project requires. Other necessary aspects are the ability to recognize words spoken by different people - in our current implementation the recognizer has only been trained with audio from a single person.

etc.

3.2. **Activity Classifier based on Words.** To infer an activity words are recognized from an audio stream collected during the activity. The collection of words that are recognized during the activity are treated as a instance to be assigned to an activity class. We formulate the problem as text document recognition problem and apply two document classification techniques for recognition. These were: (i) k-means clustering based on TF-IDF based document similarity and (ii) trained bayesian models for each activity class based on feature vectors containing words being present or not. Both of these are 'bag-of-words' models, we discuss how we use these later in the evaluation section however for our basic activity recognition had the best performance using a the bayesian model.

For both techniques we apply two steps, we first remove stop words (taken from [6]) then apply Porter's well known stemming algorithm [5]. To calculate the k-means based clustering using TF-IDF weights we use a matlab tool kit, Text to Matrix Generator [1]. To perform the bayesian analysis we use the Mallet tool [2] which is a java tool for text mining.

4. Evaluation

In this section we discuss the evaluation of the two components in this experiment, namely performance of our word recognizer and activity recognition based on spoken words.

4.1. **Word Recognition.** We initially used the HTK toolkit to build a word recognizer that recognized two words (as discussed in the milestone report). However using this toolkit abstracted away too many of the details of the recognition process so it was dumped for a simpler recognizer design but one that could be coded in C++ and run directly on the phone. We ignore the results based on the HTK toolkit and instead only talk about the performance of the C++ recognizer which runs on the iPhone.

	slices	coffee
slices	11	0
coffee	1	12

Table 2. Confusion Matrix with two word classifier. Very little background noise. Same speaker in testing as training.

	slices	coffee	stop	burrito	menu
slices	10	0	0	0	0
coffee	3	7	0	0	0
stop	0	0	10	0	0
burrito	0	0	0	10	0
menu	0	0	0	0	10

Table 3. Confusion Matrix with five word classifier. Very little background noise. Same speaker in testing as training.

	slices	coffee
slices	6	4
coffee	2	8

Table 4. Confusion Matrix with two word classifier. Music playing at reasonable background noise type of volume. Same speaker in testing as training.

We trained only on fairly clean audio from a single speaker. Two versions of the training were done: first a two word recognizer and second a five word recognizer. We used the words, 'slices' and 'coffee' in the two word version and expanded this to include 'stop', 'menu' and 'burrito' in the five word version. Training for each work was done by saying the word 10 to 15 times. Evaluation was done in a similar way, with each word being said again around 10 to 15 times, at a later time.

We present our results for the word recognizer as a series of confusion matrix. Following convention the ground truth is on the x axis and the inferences are made on the y axis.

We see low error levels for error in the two word recognizer.

To see how brittle our classifier is we perform a second experiment. One where had a different person say two words, 'slices' and 'coffee'. The volume of the background noise was set to a level that would be reasonable to still have a conversation over, but the exact volume was not measured.

The results for this experiment with background noise is promising. As seen in Table 4.1, even though only clean audio was used to train the classifier, accuracy rates of 80% and 60% (for each word) were achieved.

4.2. **Activity Recognition.** Our initial effort in doing activity recognition based on the collection of words transcribed for different examples of activities was to try and recognize a narrow set of activities that were quite specific. This recognition was attempted by using a simple k-means clustering based on TF-IDF based document similarity. The classes we identified were: *banking, going to the burrito place, going to a bookstore, a gas station, a bar, the library, the gym, a pizza place, a coffee shop*. We reported on these results in our milestone report. Of these 8 classes our average accuracy was around 15% (see Table 4.2).

We looked into this disappointing result. The main reasons were due to the fact the data was too often taken from the middle part of an activity that often looked much the same as other activities. For example

	Accuracy
Bank	0%
Boloco Burrito	0%
Dartmouth Bookstore	25%
Gas Station	0%
Murphey's Bar	15%
Library	0%
Gym	0%
Pizza Place	33%
Coffee Shop	67%

Table 5. Accuracy of classification using clustered TF-IDF approach.

	Bank	Fast Food	Coffee	Gym
Accuracy	100%	100%	81%	0%

Table 6. Accuracy of classification using bayes model.

sitting and drinking coffee is similar to sitting and eating pizza. The words used maybe conversational and hard to discriminate the classes. We realized that the start and end points of the activities, and the highly context specific conversations that occur then were the most discriminate parts of the activities. For example the interaction when leaving a restaurant or picking up a fast food item you had previously ordered.

So we collected additional data in which we made sure that the start and end points of activities were collected and transcribed as well. We also for this next stage adopted looser class definitions which were more appropriate for some of the envisioned applications. We instead adopted for four classes: *fast food, bank, coffee, gym.* These were more general and would span multiple locations, while not being overly specific (e.g., recognizing burrito places is too specific to be useful).

We then tested using bayesian models of activity based on binary indicator variables for word used as a feature. Frequency was ignored since in looking at the data it became clear that frequency was not really discriminative as much as the dictionary of words used, and in particular specific words being used. Including word frequency actually would start to identify activities in which for instance conversations were frequently compared to those activities where conversations were not (e.g., comparing being in a restaurant relative to being at the gym). We got far better results as seen in Table 4.2, which is around 71% accuracy. This result is with the test data, and with 70% ratio of test to training data.

The poorest result was for the gym class. For this class there was no successful classification. This is because all of the gym samples of audio collected contained virtually no discernible words. This was a function of how the subject that went to the gym went about these workouts - these were essentially treadmill based workouts in which the person ran for a period of time without talking to anyone. This highlights the need for this technique to be accompanied by other features (such as those based on the accelerometer) for it to be effective in a large variety of circumstances.

In Table 4.2 we see the confusion matrix of the activity classes. An important aspect of this matrix is the extra class called 'other'. Other is for a number of activities that did not fit in the classes we were interested in recognizing. This includes visits to the service station, the library to read a book, a computer lab etc. This 'other' class also deliberately includes other activities that are similar to those of the class being recognized, so that the classifier had to combat similar classes, even with the inclusion of this other

	Bank	Fast Food	Coffee	Gym	Other
Bank	4	0	0	0	0
Fast Food	0	9	0	0	0
Coffee	0	0	13	0	3
Gym	0	0	0	0	5
Other	0	1	4	0	19

Table 7. Confusion Matrix of classification using a bayes model.

class data we see little confusion which is a good sign. Example of deliberately included ‘other’ data for instance including visits to restaurants (that maybe confused with fast food).

We find classification works so well because the each example of an activity has a number of very distinctive words that identify it from the others. For example in the pizza places the keywords ore things like ‘large’, ‘pizza’, ‘cheese’. In the coffee shops we observe words that are most discriminative are those such as ’tea’, ’coffee’ and ‘macchiato’,

Our most promising results are the ability for this process to solve very difficult examples of human activity recognition. For example, we find the classifier is able to:

(1) Identify examples of the same class, even though the physical location had changed and the subtype of the activity had changed. A good example of this is our result for the fast food class. It was able to recognize examples of purchasing burritos, as well as purchasing pizza. Even though both are in different physical locations and the activities are fairly distinct from each other (burritos versus pizza). The binding aspects is that both are part of the same underling class.
(2) Recognize different uses of the same physical space. Location is relied upon a lot to determine an activity. However it is hard to recognize when the same location have different possible activities. An example of this is the Dartmouth bookstore. The classifier is able to separate visits to the bookstore to purchase books relative to purchasing coffee.

-

5. Future Work and Conclusions

The results here in the preliminary experiments are promising. I would like to improve this work by expanding to include other features beyond only the microphone. I think there are likely other examples of the more general idea of leveraging features where humans already distill a great deal of data for the classifier - versus features attempting to do all the work itself. Other possible examples include: the number of words, the emotional content of the words, characteristics of conversation by the person.

Clearly one aspect of this project would be to complete the entire system and perform complete experiments, this would require a word recognizer which is much more robust to the noise. With a completed system definitely additional work evaluating the system, such as collecting a more diverse set of activities and sounds, is required.

Right now the system is quite brittle and tied to particular words that are recognized. Although it works very well, on the data collected, the question is low well would it work with a much different set of activities in a different location. There is not enough data to provide any conclusive results. One particular problem would be those activities that are of the same class but for which the vocabulary used is different. For instance, two coffee shops may have words such as ‘tea‘, ‘chai’ while another may not have have only words like ‘coffee’ , ‘beans’. One approach to this would be to use an additional large corpus of terms, and look for co-occurrence between words. If this corpus suggests that coffee and tea are linked because they co-occur with high frequency. Then the two activities could be recognized as being similar due to this the shared co-occurance between words they each have in their document vector. Techniques similar to this have been proposed before in text retrieval [11].

References

[1] Text to Matrix Generator. http://scgroup20.ceid.upatras.gr:8000/tmg/index.php/Main_Page
[2] Mallet. http://mallet.cs.umass.edu/
[3] VQ implementation http://www.data-compression.com/vq.shtml
[4] HMM source files used http://leb.net/pub/blinux/hmm-1.03.tar.gz
[5] Porters Stem Algorithm http://tartarus.org/~martin/PorterStemmer/java.txt
[6] Stopword List http://www.textfixer.com/resources/common-english-words.txt
[7] D. Zaykovskiy Survey of the Speech Recognition Techniques for Mobile Devices
[8] C.J. Rijsbergen Information Retrieval
[9] Training Hidden Markov Model/Artificial Neural Network (HMM/ANN) Hybrids for Automatic Speech Recognition (ASR). http://speech.bme.ogi.edu/tutordemos/nnet_training/tutorial.html
[10] B. ..Gold, N. .Morgan, N. Nelson Morgan3 Speech and Audio Signal Processing: Processing and Perception of Speech and Music. Ben Gold, Nelson Morgan
[11] A. Huang, B David Milne, E .ibe Frank, and Ian H. Witten Clustering Documents using a Wikipedia-based Concept Representation