CS 288: Speech Recognition Decoding

Anting Shen
23566738
`antingshen@berkeley.edu`

1 Introduction

2 Data Structures

The pronunciation dictionary is stored as a prefix trie. Each node in the trie contains the phoneme at the node, a list of child nodes, a list of words that are made of all phonemes up to and including this node, and the max unigram probability of all words that are in the subtree rooted at this node. The nodes also contain parent pointers for convenience. When the trie is built when the recognizer is intialized, all words whose subphones are not all in the acoustic model are ignored. When backoff to non-contexual subphones were used instead of discarding the words altogether, the results were worse, most likely due to unseen words being rare and bad anyway.

The states stored for the HMM include the previous two words for the trigram LM, a pointer to the lexicon trie node, a pointer to the previous state, the subphone at this state, and the state's probability. Backpointers to previous states means that only one beam needs to be stored to recover prediction and eliminated hypotheses will be garbage collected.

This decoder uses beam search where each beam is a priority queue of State objects. Beams are implemented as primitive binary min-heaps, with a HashMap alongside it that maps from State to its location in the heap. The HashMap is used for recombination. The beam supports `relax` and `max`. `relax` adds a state to the beam if no equivalent state is present, and keeps the state with higher probability otherwise. In the event that the state in the beam has lower probability, its probability is increased and the state is bubbled down in the heap. The duplicate state is always discarded. When bubbling up or down, the HashMap values are kept in sync when states are moved, and a hole is bubbled through the heap until the end instead of moving the state each stage to minimize HashMap accesses. `max` gets the state with the highest probability and is not optimized since it's only called once at the end of the utterance.

3 Decoding Process

The beam is initialized with a state for each possible starting phone in the lexicon at its first subphone. Then for every timestep, it creates a new beam and places every transition of every hypothesis of the old beam into the new beam, and the beam will keep the best ones.

Every state has the transition of self looping into an identical state except with a probability of the previous probability plus $P(\text{mfcc}|\text{subphone})$.

Each state can also transition into the next subphone, with the appropriate backward and forward contexts. Those transitioning from subphone 2 to 3 will have a transition for each possible next phoneme in the lexicon because the next phoneme must be decided at the time the forward context is needed. If a word ends at this phoneme, another transition is added with no forward context. A LM-smearing unigram cost difference between the previous phoneme and the current one is paid during this transition as well.

When transitioning out of a word, the unigram cost is replaced by a trigram LM probability, and at the beginning of a sentence, start tokens are added to the front, though this seems to have no effect on WER. All possible starting phoneme states are then added to the beam with prevWord pointers updated.

After all the feature vectors are processed, the best state is found in the last beam and the prediction is extracted from following backpointers.

4 Accuracy Optimizations

The recognizer uses a language model scaling factor of 8, and a word bonus of $log(1.1)$. The unigram LM is also smeared across phonemes. These are implemented in fairly standard ways.

Much of the effort went into optimizing the recognition of the ends of utterances. The original algorithm often ends up with final beams filled with hypotheses that haven't quite finished the last word, since the hypotheses that did not finish the last word haven't paid the trigram word penalty. This causes the prediction to mispredict or omit the last word.

At first, I attempted to simply add the word that the best state in the last beam was in the middle of to the end of the prediction, but there were no improvements in the WER since the last word is often incorrect. Instead, the beam size is increased towards the end of the utterance, up to four times the base size for the last two MFCC's. Additionally, to encourage the transition out of a word, the word bonus is multiplied by a multiplier given by $max(1, 10/(num\_left + 0.03))$, for a 30x bonus on the last iteration.