Subtitle Generation for Videos

Utkarsh Pathrabe (2012A7PS034P)

Nitin Labhishetty (2012A7PS038P)

Pavan R (2012C6PS331P)

# Problem & Motivation:

Every time we watch a movie we have to search for subtitles, download them and add it manually to the video player. Further, if the sync and quality aren't good, we have to repeat the process. And for other videos such as lectures, conference or informational talks we seldom find subtitles. If the subtitles can be made from the audio content of the videos, it will be very convenient for the users.

# Brief code walkthrough:

1. Extracting the audio from videos using JAVE. It is a Java library that is a wrapper around ffmpeg. VideoToAudio class handles the conversion process.

Brief working:

1. Open source and destination files
2. Set AudioAttributes for the output file. Program set to encode audio in 16 kHz, 16 bit mono wav format.
3. Encoder class used for extracting audio from source with AudioAttributes into destination file. Encode () in encoder class takes these 3 parameters and does the extraction procedure.
4. Extracting speech from audio using Sphinx4 Java library. Transcriber class handles audio to text conversion. Brief working:
5. Every speech recognition process requires 4 attributes- the dictionary, language model, acoustic model and speech source.
6. Set dictionary, language model and acoustic model as described in Dataset description.
7. StreamSpeechRecognizer class takes in audio source from a stream for recognition. StreamSpeechRecognizer object is made with earlier set configuration.
8. SpeechResult container is used for holding result of recognition performed by StreamSpeechRecognizer object. The recognition is performed till no more result is obtained from input stream.
9. SpeechResult gives words along with the timestamp in a format used by subtitle files.

# Dataset and trained models used:

There are 3 trained models required for Audio to text translation:

1. **Language Model** - It comprises of

* Language model: Language model used is en-us.lm.dmp. It is created by 13000 words from WSJ (Wall Street Journal) text after pre-processing the text by converting numbers to words, short forms to full forms and punctuation marks being removed. .dmp format has data in binary format which makes loading of trained models very fast.
* Grammar model: Structure of words according to English grammar rules is specified in this model. E.g. (Good morning | Hello) (Bhiksha | Evandro | Paul | Philip | Rita | Will) implies that Good morning | Hello precedes Biksha | Evandro | Paul | Philip | Rita | Will. For the large vocabulary the search space is configured differently. Grammars are not effective to describe very big word lists. Instead, n-gram language models are used. This is specified in alphabet.arpaformat file.

1. **Dictionary Model** – English language already has a carefully prepared language dictionary which is fine tuned for best performance. It comprises of alpha.dict, digits.dict and cmudict.0.6d files.

* Alphabet Dictionary (alpha.dict): It comprises of letter as key and set of phonemes as the value corresponding to the key. There can be more than one phoneme for a particular letter.

E.g. (A EY), (B B I Y), (B (2) B IH)

* Digit Dictionary (digits.dict): Digits are represented as text and not as numbers. Dictionary comprises of numbers as keys and corresponding set of phonemes as values. There can be more than one phoneme for a particular digit.

E.g. (ONE HH W AH N), (ONE (2) W AH N)

* Word Dictionary (cmudict.0.6d): It comprises of a word as key and pronounced spelling of word as value. 13000 key value pairs are specified in cmudict.0.6d.

E.g. (ABRAMO AA B R AA M OW), (ACCENT AE K S EH N T)

3. **Acoustic Model** - US English acoustic models for microphone is prepared using Wall Street Journal by varying number of senones, continuous/non-continuous attribute, HMM topology and number of gaussians per state. Acoustic model used for project is WSJ\_8gau\_13dCep\_16k\_40mel\_130Hz\_6800Hz which can be downloaded from [*https://code.google.com/p/motenav/downloads/detail?name=WSJ\_8gau\_13dCep\_16k\_40mel\_130Hz\_6800Hz-1.12\_r8.jar&can=2&q=*](https://code.google.com/p/motenav/downloads/detail?name=WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz-1.12_r8.jar&can=2&q=)

# Some theory behind the recognition process:

Speech recognition performed has the following components:

* + - 1. Classifying Audio segments to phonetic sounds in a language. The target set of phonetic sounds in a language are specified by the Acoustic model.
      2. Combining phonetic sounds to words. This is done by using Dictionary model. HMM is used for time series analysis of phonetic sounds to recognize set of target words.
      3. Forming sentence from words through semantic analysis. This is done using Language model. Every sequence of phonetic sounds can result in a set of words e.g. odyssey | ought to see | O to C, all these sounds similar, one may be more appropriate than another based on context.



Recognition of phonetic sounds called as phonemes into words

Hidden Markov Models (HMMs) provide a simple and effective framework for modelling time-varying spectral vector sequences. As a consequence, almost all present day large vocabulary continuous speech recognition (LVCSR) systems are based on HMMs.



The principal components of a large vocabulary continuous speech recogniser are illustrated in Figure above. The input audio waveform from a microphone is converted into a sequence of fixed size acoustic vectors in a process called feature extraction. The decoder then attempts to find the sequence of words which is most likely to have generated the feature vector. However, since *P*(***word****|****Feature vector*** ) is difficult to model directly, Bayes Rule is used.

The likelihood *p*(***Feature vector*** *|****word***) is determined by an *acoustic model* and the prior *P*(***word***) is determined by a *language model*. The basic unit of sound represented by the acoustic model is the *phone*. For example, the word "bat" is composed of three phonemes /b/ /ae/ /t/. About 40 such phonemes are required for English. For any given ***word***, the corresponding acoustic model is synthesized by concatenating phone models to make words as defined by a pronunciation dictionary. The parameters of these phone models are estimated from training data consisting of speech waveforms and their transcriptions. The language model is typically an *N*-gram model in which the probability of each word is conditioned only on its *N −* 1 predecessors. The *N*-gram parameters are estimated by counting *N*-tuples in appropriate text corpora. The decoder operates by searching through all possible word sequences using pruning to remove unlikely hypotheses thereby keeping the search tractable. When the end of the utterance is reached, the most likely word sequence is output. Alternatively, modern decoders can generate lattices containing a compact representation of the most likely hypotheses.

# Requirements for running programs:

1. Video to audio requires JAVE library, steps to set this up are:

* Download JAVE source 1.0.2 from http://www.sauronsoftware.it/projects/jave/download.php and add all JAVA classes in it to your eclipse package.
* Go to temp folder (start->run->(type) %temp%)
* Create new folder jave-1
* Add ffmpeg application and pthreadGC2.dll to jave-1. Both files are found in java 1.0.2 folder.

1. Audio to text requires sphinx setup as follows:

* Download sphinx4 from <http://sourceforge.net/projects/cmusphinx/files/sphinx4/5%20prealpha/>
* Go to File > Import > Existing Project into Workspace and choose the sphinx4 zip file downloaded in the previous step.
* Right click on build.xml file in the project and choose run as Ant build.

In case of missing jar errors, add jar files manually from extracted content of sphinx4 zip folder