Supplementary files

Part 1 - Algorithm

***Algorithm 1*– Speech to Text conversion process using HMM**

Input: *Textual data whose patterns need to be converted into speech signals by applying the Hidden Markov Model procedure. The text series is the unknown states whose patterns need to be identified and make it a known state*

Output: *Speech of a Known language*

1: Begin

2: For a given text of some arbitrary language, check if the language is English or multi-lingual with one language English. If so, go to step 3 else exit

3: Perform Tokenization for every word segments in the given English Language

4: For every token, perform the following

5: Start of the loop

6: Every token is placed as a separate frame

7: In each frame, create a bi-gram model for every syllable combination

8: Every bi-gram model is now considered a window

9: End loop

10: End for

11: For every window of the frame

12: Start of the loop

13: Every bi-gram syllable combination is now checked across the chosen phonetic dictionary. In this case – Emu and Buckeye Speech corpuses

14: The given bi-gram will have specific matches from the Speech corpora. If the match is only one, then it is considered final otherwise, out of all matches found, calculate the probability of every phone utterance for a given bi-gram.

15: Evaluate the probability using HMM model. The match with the highest probability wins the race, and the corresponding phonemes are retrieved from the dictionary

16: End for

17: End of loop

18: Once again, perform HMM training to club all the phonemes to form individual words out of each window

19: After every word boundary is reached, a silence of 0.05 seconds is given

20: All the frames are concatenated together along with the word boundary buffer duration to from a sentence or a phrase

21: Arrive at the speech of the known language (English)

22: End

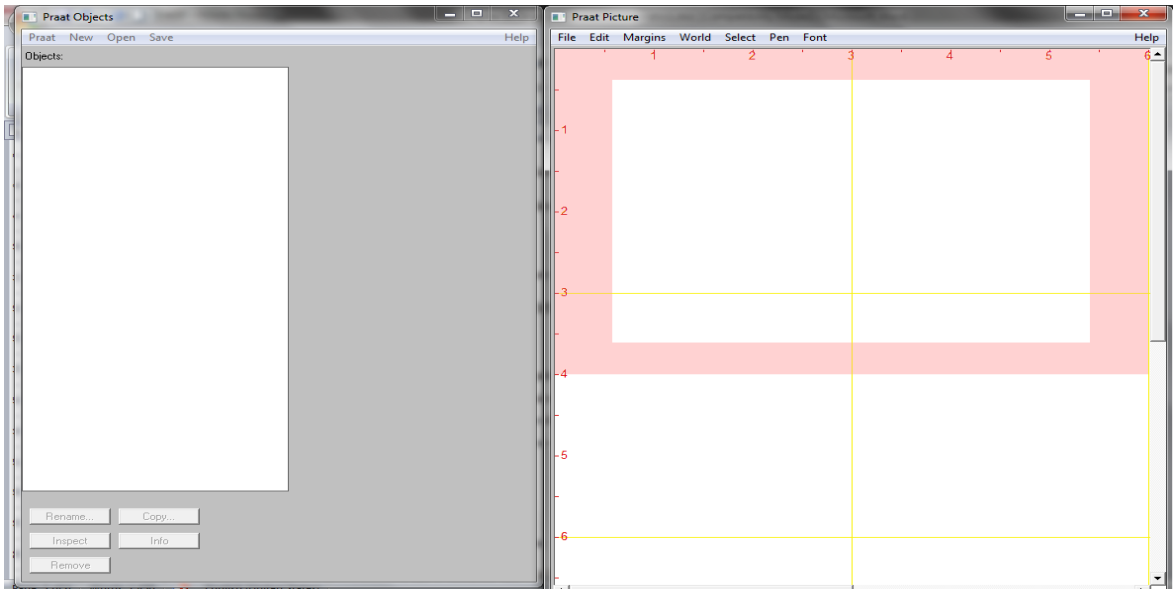


Figure A – A Praat Window showing the Praat Object Panel and Praat Picture window

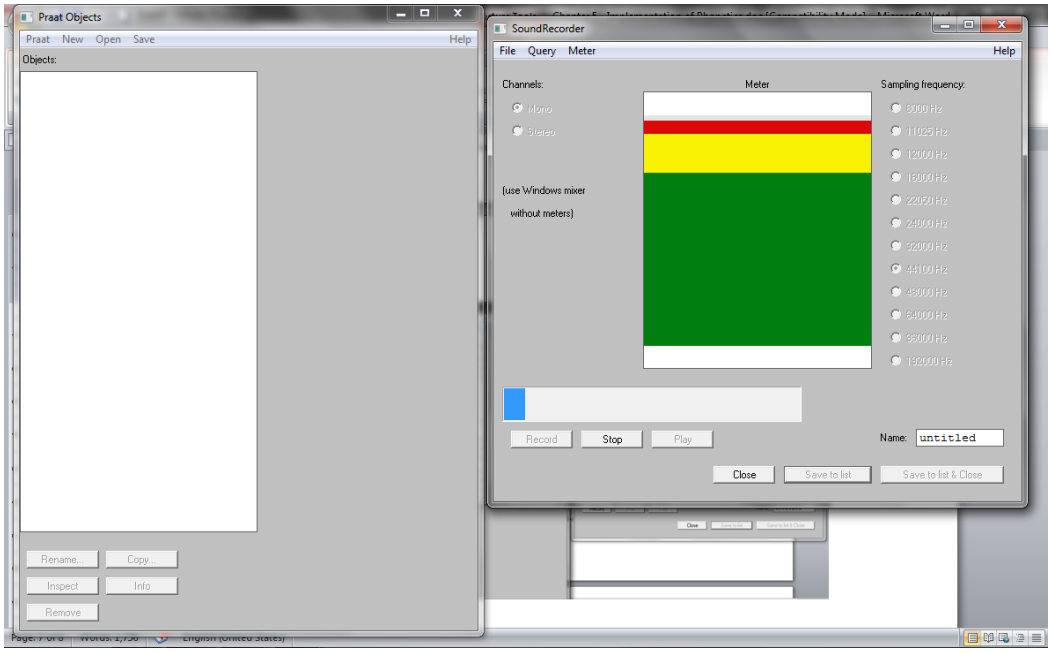


Figure B– The process of sound recording in Praat

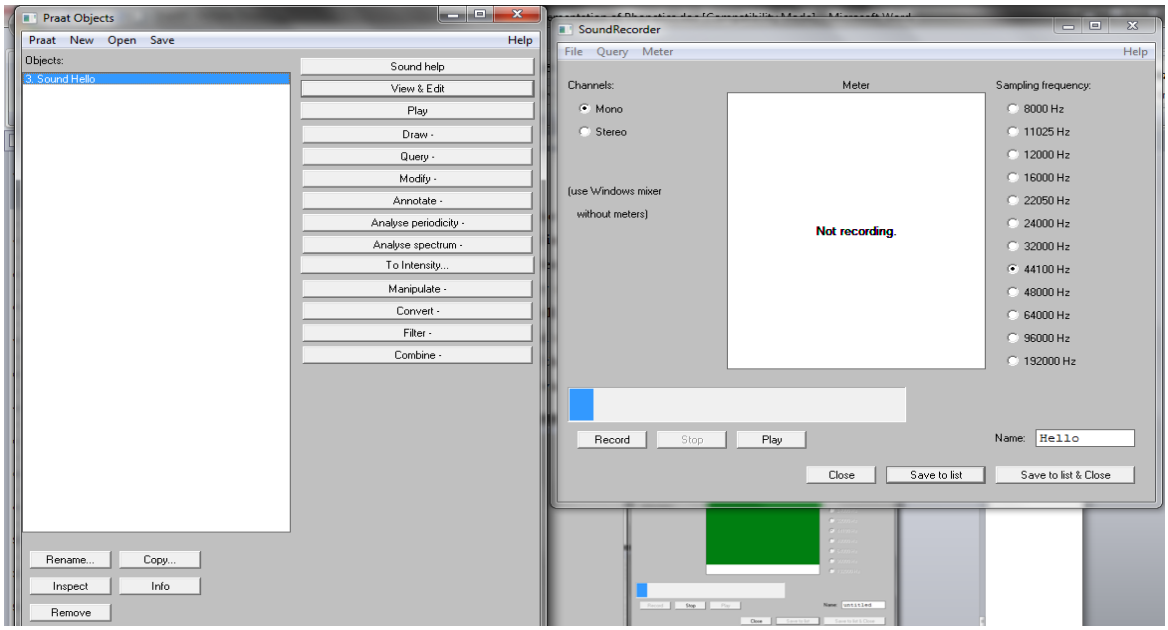


Figure C – Identification of recorded sound on the left side of the panel and the various options on the right side of the Praat objects available to analyze the recorded sample.

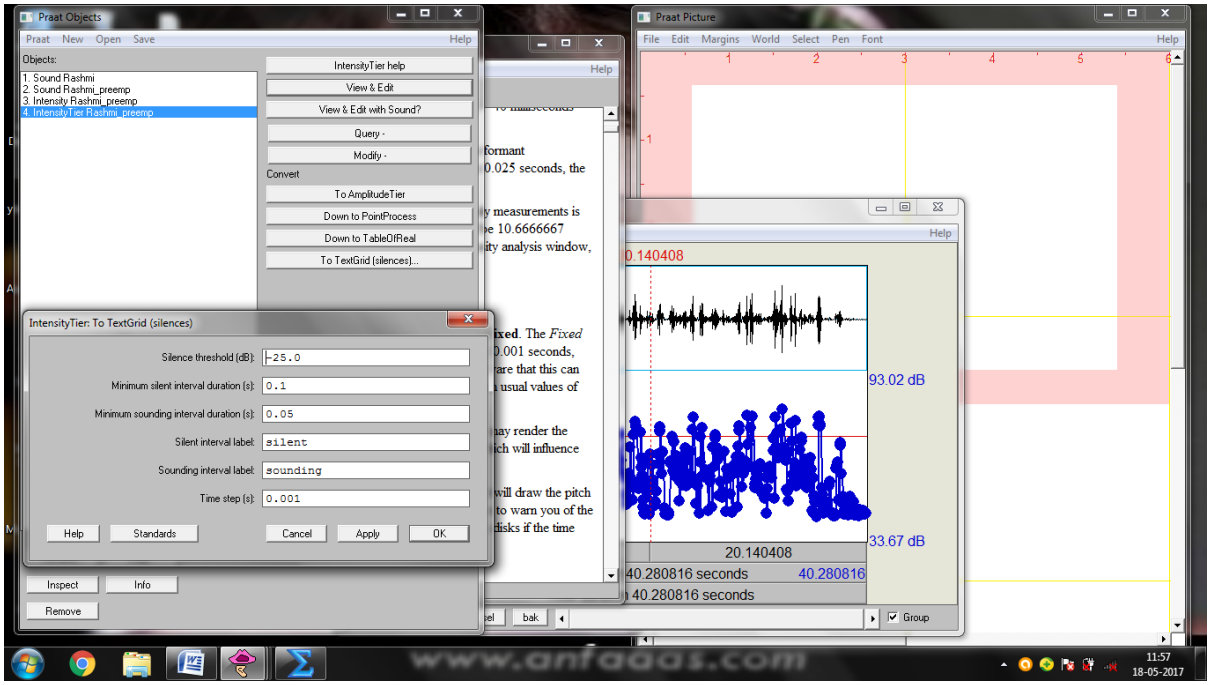


Figure D – Process of setting intensity parameters for recognition of minimum sound and silent boundary.

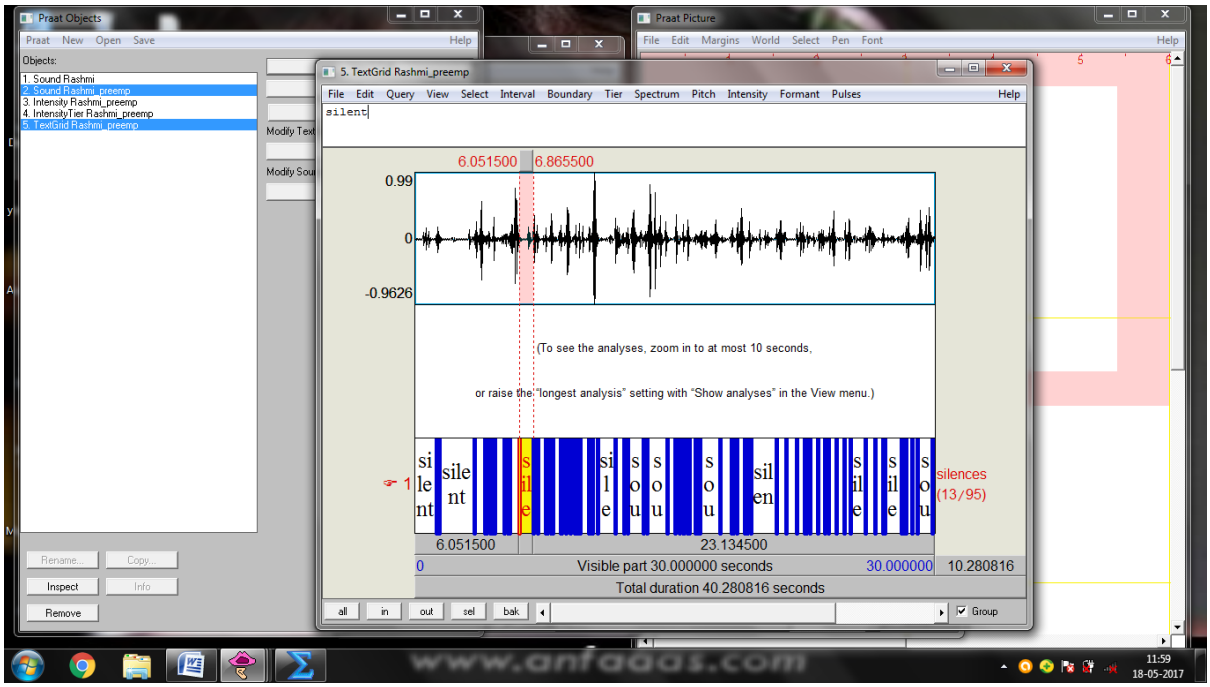


Figure E – Final output of speech pause detection process. The figure shows the separation of silent segments and sounding elements.

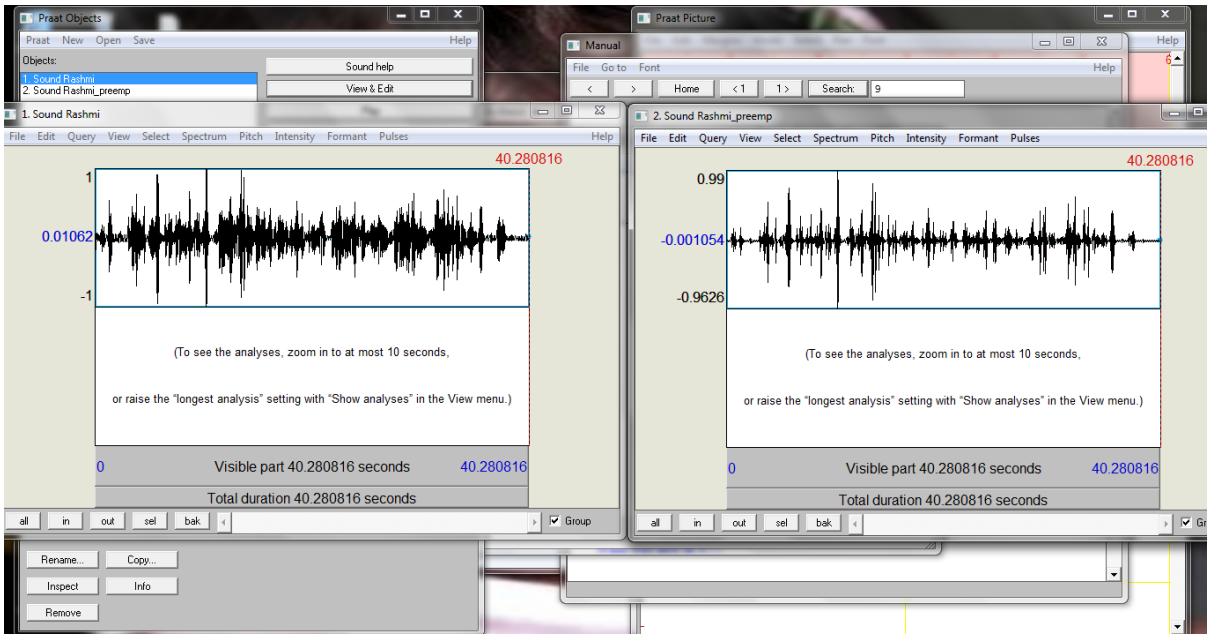


Figure F – Audio sample – Before Pre-emphasis and After Pre-emphasis

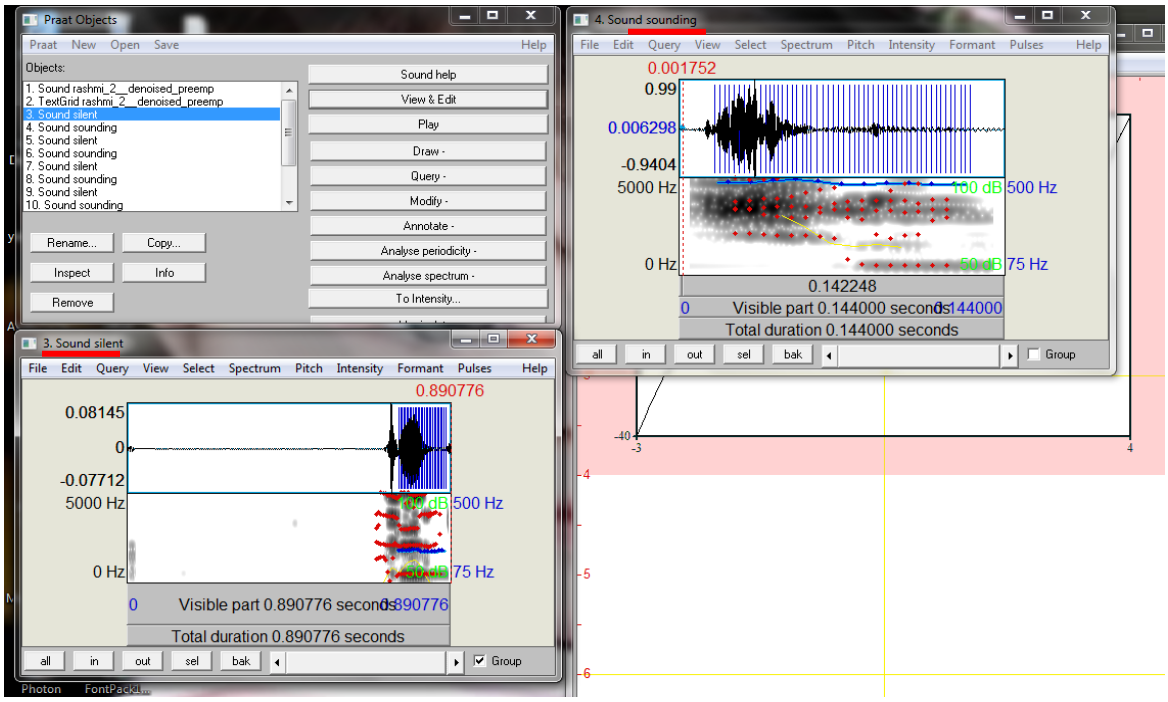


Figure G – Spectral features of a sound element and silent element separated as frames, respectively. Each of these is marked with a red color line.

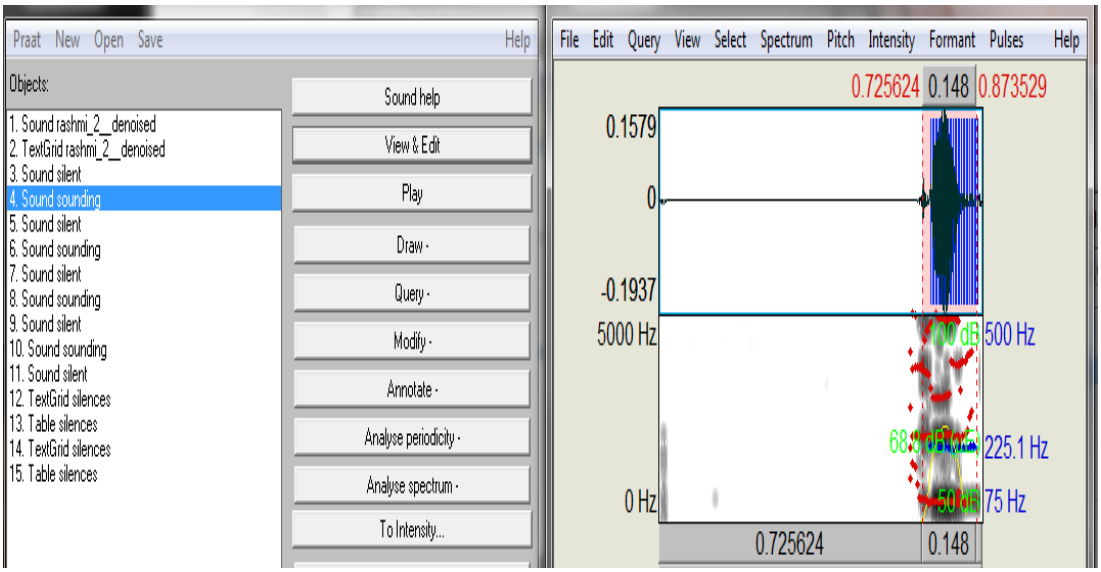


Figure H: The total duration of the features present in the first frame "the" is 0.148sec.

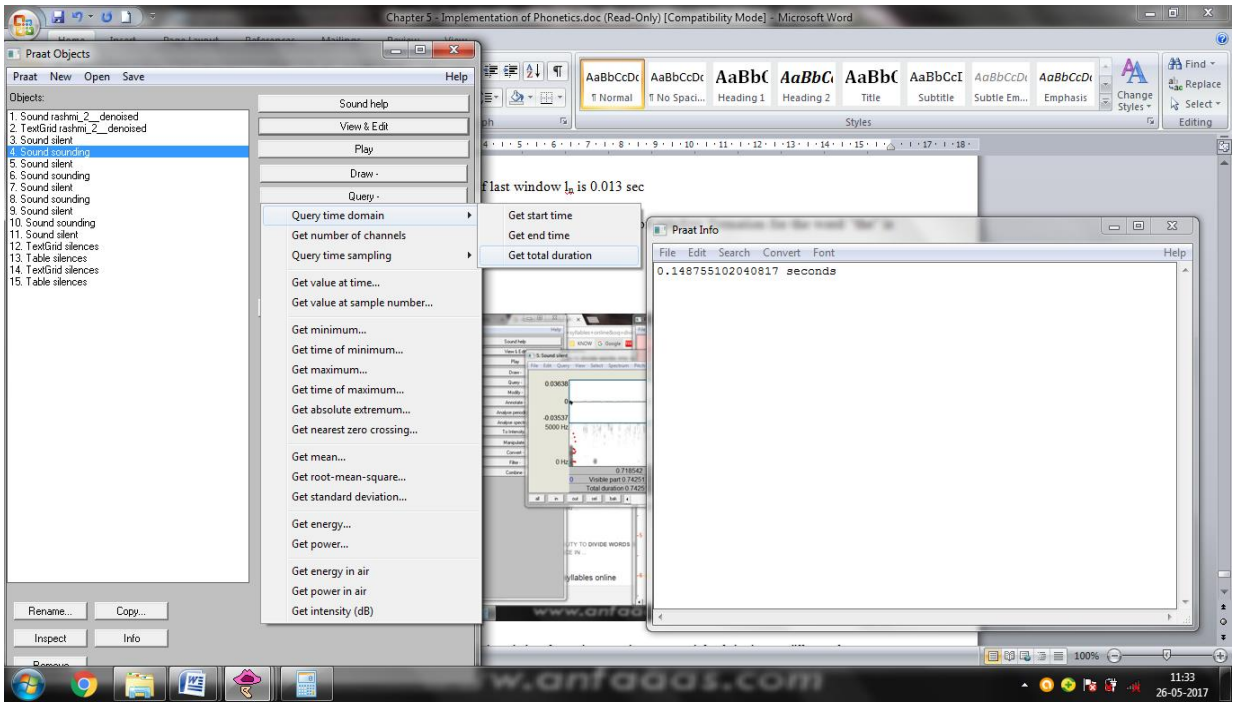


Figure 11 – The total duration of the word "the" in the first frame is shown. To know the time, select the frame and choose the option query on the right panel of the window. Under the query time domain, get total duration is selected, and the Praat info will contain the total duration of the word present in that particular frame.

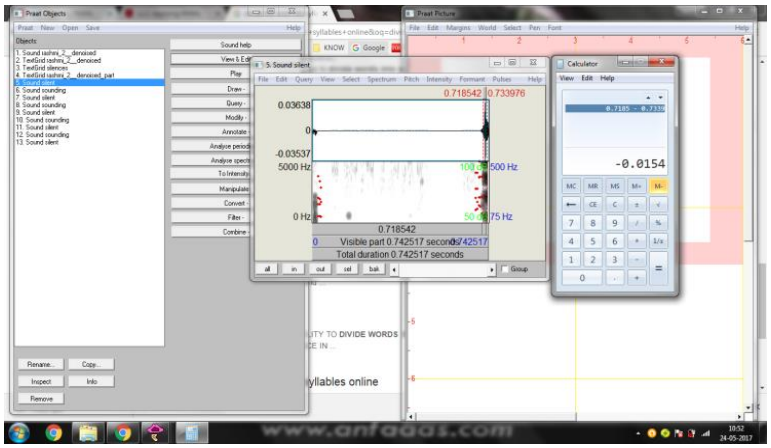


Figure I: Screenshot of the window formation in Praat. "the" has been split to the first window (W1) of length 0.015 sec. The reading 0.733 – 0.718 gives the size of this window = 0.0154

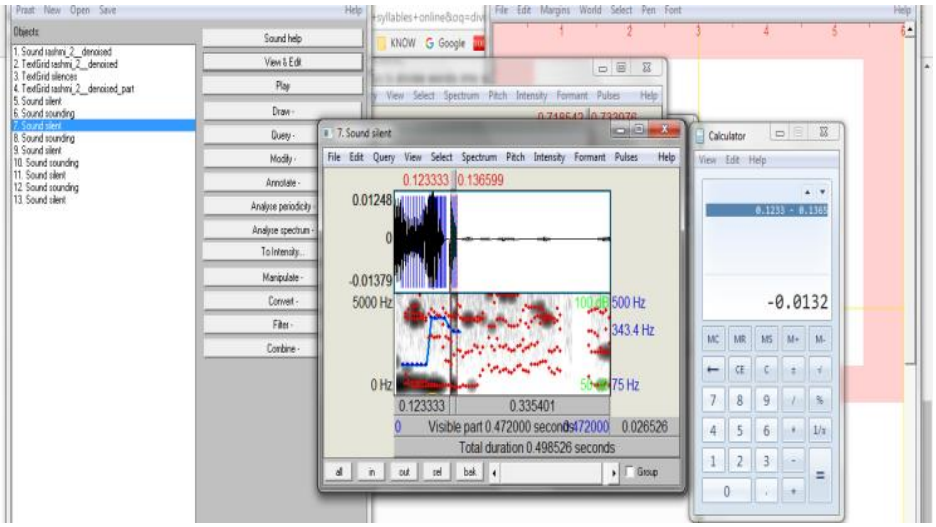


Figure J: The last window formation of "the" last window (Wn) of length 0.013 (0.136-0.123)

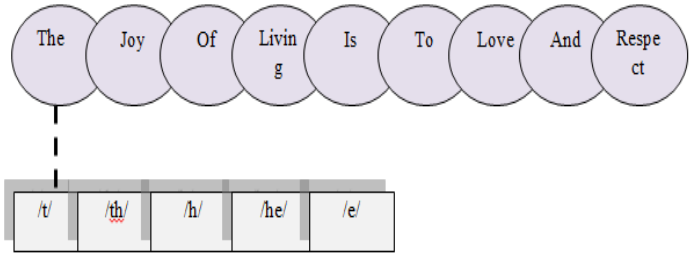


Figure K: Formation of frames and windows through HMM with the following significance; 1) Spoken utterance 2) formation of frames with an overlap region of 5ms 3) formation of a window for one frame with an overlap region of 0.05ms

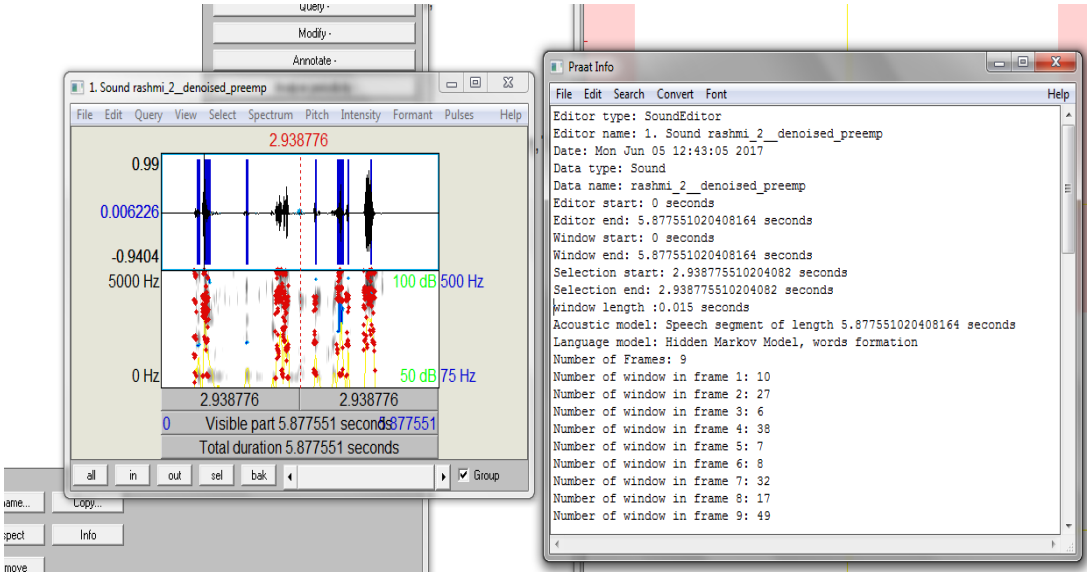


Figure L: The acoustic feature is represented on the left side. On the right side, the input is evaluated into various features resulting in nine frames, each containing a few windows.

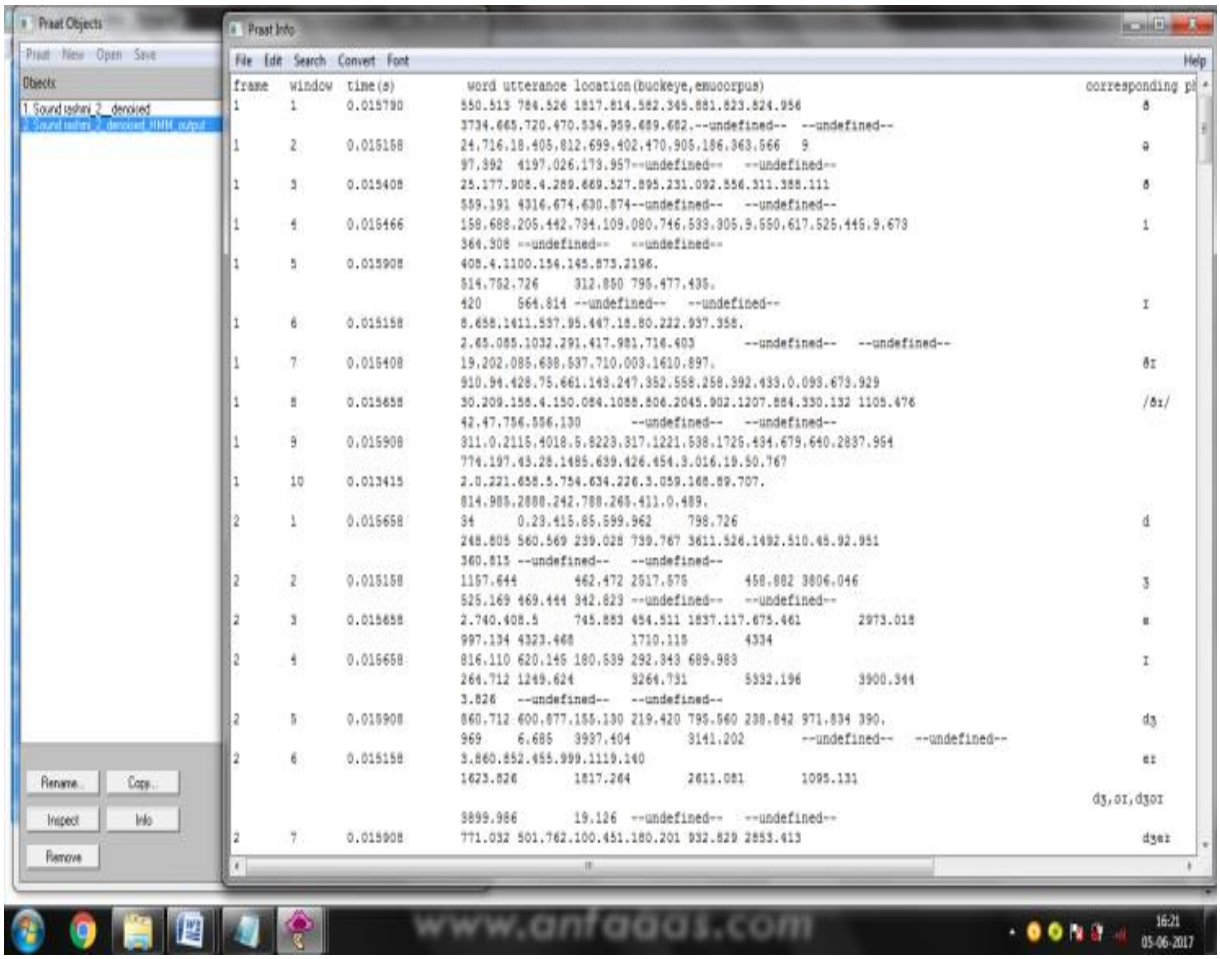


Figure M: An output showing a complete set of frames, windows in each frame, location where the window's content is present in the corpus, corresponding phonemes after applying HMM, and finally word extraction for the corresponding phoneme mapping.

Fig M illustrates the following specifications

- At first, the given sound sample is divided into a specified number of frames through MFCC. The given example is divided into nine sounding elements as, “the”, “joy”, “of”, “living”, “is”, “to”, “love”, “and”, &“respect” respectively, shown in **fig L.** These frames are indicated as a number ranging from 1 – 9 in **fig M.**
- Secondly, each frame contains a specified number of windows, **fig L.** The number of windows for a given frame is dependent on the time duration of speech utterance belonging to that particular frame. The window length is kept constant as 0.015 seconds, **fig M**. However, every frame’s last window will have a varying size calculated by the windowing procedure
- HMM is used to extract the acoustic features and map them to a corresponding phoneme representation. Further, the phonemes are searched for their corresponding match in the speech corpora, **figure M.**

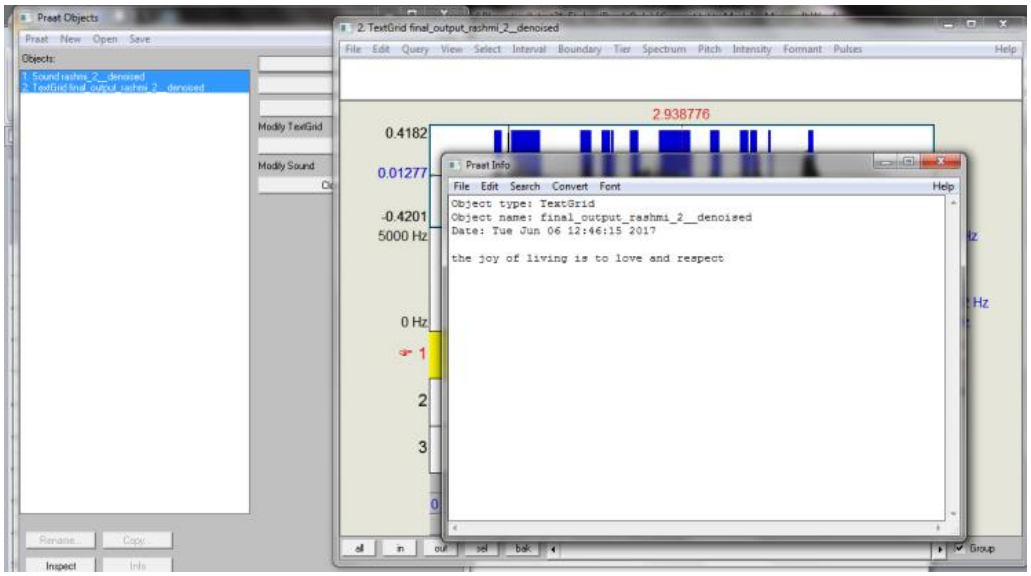


Figure N: Projecting the final output of the proposed system. A given input sound file is converted into the corresponding text. the match against each phoneme is retrieved from the corpus, and DTW is applied to concatenate the final output.

Part II:

Detailed explanation about each window. Here the probability of individual syllable is obtained by dividing it by the total number of words in the corpus containing that syllable

- **The**

Words containing‘t’ pronunciation in the corpus = 87

P(t|0,0) = Probability of ‘t’ coming first = 31= 0.35 (31/87)

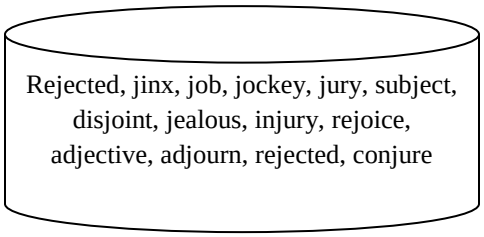
P(h|t,0) = Probability of ‘h’ coming after ‘t’ at the beginning = 19 = 0.21(19/87)

P(e|h,t) = Probability of ‘e’ coming after ‘th’ = 29 = 0.33(29/87)

Therefore each of the phoneme of the word ‘the’ →ðə,ðɪ,ðiː/ is now transformed into its corresponding syllable

Using the pronunciation of ‘the’ as trained data, more words containing ‘the’ sequence such as ‘this, there, these, then, and thesis’ are tested. These words are correctly recognized by the speech model and have been successfully converted to the exact match of syllable representation.

- **Joy**



Words containing ‘j’ pronunciation in the corpus = 14

P(j|0,0) = Probability of ‘j’ coming first = 5 = 0.38(5/14)

P(o|j,0) = Probability of ‘o’ coming after ‘j’ at the beginning = 2 = 0.15(2/14)

P(y|o,j) = Probability of ‘y’ coming after ‘jo’ = **0**

“joy” pattern of word utterance was not found in the speech corpus. Therefore with the help of HMM, the given phonemes are split into 2 different probabilities as follows;

1) ‘jo’ →P(o|j,ω) here ‘ω’ indicates any sequence of phonemes. This will lead to probability of ‘o’ coming after ‘j’, i.e., ‘2’ + any other words in the dictionary with simply ‘jo’ combination. The words ‘rejoice’ and ‘adjourn’ are found in the dictionary suiting this criterion. Therefore the total probability will be 2+2 = 4 = 0.26 (4/14)

2) ‘oy’ →P(y|o,ω) here ‘ω’ indicates any sequence of phonemes. When searched in the corpus, the phoneme for ‘oy’ was found in the pronunciation of the word ‘annoy’. Therefore the probability will become 1 = 0.07(1/14)

Supposedly, ‘joω’ was not found and ‘ωoy’ was not found then the HMM model will look for ‘ωjω’ alone and the same with ‘ωoω’ and ‘ωyω’

Therefore each of the phoneme of the word ‘joy’ →dʒɔɪ/ is now transformed into its corresponding syllable

- **Of**

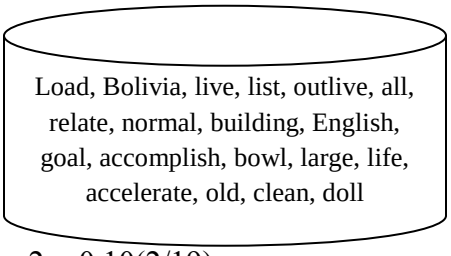
Words containing ‘o’ pronunciation in the corpus = 13

P(o|0,0) = Probability of ‘o’ coming first = 3 = 0.28(3/13) → object, of, office

P(f|o,0) = Probability of ‘f’ coming after ‘o’ at the beginning = 2 = 0.17 (2/13)→ office, of

Therefore each of the phoneme of the word ‘of’ →ɒv,(ə)v/ is now transformed into its corresponding syllable

- **Living**



Words containing ‘l’ pronunciation in the corpus = 19

P(l|0,0) = Probability of ‘l’ coming first = 5 = 0.26(5/19)

P(i|l,0) = Probability of ‘i’ coming after ‘l’ at the beginning = 2 = 0.10(2/19)

P(v|i,l) = Probability of ‘v’ coming after ‘li’ = 3 = 0.15(3/19)

P(i|v,i) = Probability of ‘i’ coming after ‘ivi’ = 0

“ivi” pattern of word utterance was not found in the speech corpus. Therefore with the help of HMM, the given phonemes are split into 2 different probabilities as follows;

1) ‘iv’ →P(v|i,ω) here ‘ω’ indicates any sequence of phonemes. This will lead to probability of ‘v’ coming after ‘i’, i.e., ‘3’ + any other words in the dictionary with simply ‘iv’ combination. Words ‘active’ and ‘divine’ were found in the dictionary. Therefore the total probability will become 3+2 = 5 = 0.26(5/19)

2) ‘vi’ →P(i|v,ω) here ‘ω’ indicates any sequence of phonemes. When searched in the corpus, the phoneme for ‘vi’ was found in the pronunciation of the words ‘behaviour’ and divine. Therefore the probability will become 2 = 0.10(2/19)

Supposedly, ‘ivω’ was not found and ‘ωvi’ was not found then the HMM model will look for ‘ωiω’ alone and the same with ‘ωvω’ and ‘ωiω’

P(n|i,v) = Probability of ‘n’ coming after ‘vi’ = 1 = 0.05(1/19) →convince

P(g|n,i) = Probability of ‘g’ coming after ‘in’ = 4 = 0.21(4/19) →building, seeing, drawing, caring

Therefore each of the phoneme of the word ‘living’ →ˈlɪvɪŋ/ is now transformed into its corresponding syllable

- **is**

Words containing ‘i’ pronunciation in the corpus = 114

P(i|0,0) = Probability of ‘i’ coming first = 7 = 0.06(7/114) → illustrate, island, ignore, is, idea, identify, important, and if.
P(s|i,0) = Probability of ‘s’ coming after ‘i’ at the beginning = 2 = 0.01(2/114) → is and island.

Therefore each of the phonemes of the word ‘of’ →IZ/ is now transformed into its corresponding syllable
In the later stages, these phoneme patterns will be converted into text form after applying the dynamic time wrapping procedure.

- **To**

Words containing ‘t’ pronunciation in the corpus = 87

P(t|0,0) = Probability of ‘t’ coming first = 31= 0.35(31/87)
P(o|t,0) = Probability of ‘o’ coming after ‘t’ at the beginning = 5 = 0.05(5/87) →total, together, to, tour, touch

Therefore each of the phoneme of the word ‘of’ →tə,tʊ,tuː/ is now transformed into its corresponding syllable

- **Love**

Words containing ‘l’ pronunciation in the corpus = 19

P(l|0,0) = Probability of ‘l’ coming first = 5 = 0.26 (5/19)→life, large, live, list and load
P(o|l,0) = Probability of ‘l’ coming after ‘t’ at the beginning = 1 = 0.05 (1/19)
P(v|o,l) = Probability of ‘v’ coming after ‘lo’ = 0
“lov” pattern of word utterance was not found in the speech corpus. Therefore with the help of HMM, the given phonemes are split into 2 different probabilities as follows;

1) ‘lo’ →P(o|l,ω) here ‘ω’ indicates any sequence of phonemes. This will lead to the probability of ‘o’ coming after ‘l’, i.e., ‘1’ + any other words in the dictionary with simply ‘lo’ combination. No other words apart from ‘load’ were found in the dictionary. Therefore the probability becomes 1 = 0.05(1/19)

2) 'ov' →P(v|o,ω) here 'ω' indicates any sequence of phonemes. This is the probability of 'v' coming after 'o'. When searched in the corpus, the phoneme for 'ov' was found in the pronunciation of the words 'above' and 'approve'. Therefore the probability will become 2 = 0.10(2/19)

Supposedly, 'loω' was not found and 'ωov' was not found then the HMM model will look for 'ωlω' alone and the same with 'ωoω' and 'ωvω'

P(e|v,o) = Probability of 'e' coming after 'ov' = 2 = 0.10(2/19)→ above and approve

Therefore each of the phoneme of the word 'love' →ˈlʌv/ is now transformed into its corresponding syllable

- **And**

Words containing 'a' pronunciation in the corpus = 208

P(a|0,0) = Probability of 't' coming first = 67 = 0.32(67/208)
P(n|a,0) = Probability of 'o' coming after 'a' at the beginning = 23 = 0.11(23/208)
P(d|n,a) = Probability of 'd' coming after 'an' = 6 = 0.02 (6/208)→demand, abandon, and, android, bandage, and candle.

Therefore each of the phonemes of the word 'and' →ənd is now transformed into its corresponding syllable

- **Respect**

Words containing 'r' pronunciation in the corpus = 82

P(r|0,0) = Probability of 'r' coming first = 13 = 0.15(13/82)
P(e|r,0) = Probability of 'e' coming after 'r' at the beginning = 3 = 0.03(3/82) → react, reality and recall
P(s|e,r) = Probability of 's' coming after 're' = 2 = 0.02(2/82)→address and arrest
P(p|s,e) = Probability of 'p' coming after 'es' = 2 = 0. 02(2/82)→Desperate and cheese
P(e|p,s) = Probability of 'e' coming after 'sp' = 1 = 0.01(1/82) →Desperate
P(c|e,p) = Probability of 'c' coming after 'pe' = 1 = 0.01(1/82)→Suspect
P(t|c,e) = Probability of 't' coming after 'ec' = 1 = 0.01(1/82)→object

Therefore each of the phonemes of the word ‘respect’ →rɪˈspɛkt/ is now transformed into its corresponding syllable

Every time the phonemes are mapped between the windows, a dynamic text wrapping procedure is applied. Dynamic text wrapping is a variation of dynamic time wrapping. Dynamic text wrapping is a process where the words between the windows are wrapped and merged after applying HMM. Since the text is being wrapped between the windows, it is called dynamic text wrapping. If speech or acoustic features are being wrapped as in the Text-to-Speech model, then it is called dynamic time wrapping since the speech features are wrapped between the windows without losing any second of data.

Part III

Sample results obtained for the chosen example in Speech to Text conversion

framewindow time(s) word utterance location(buckeye,emucorpus) corresponding phoneme word generation

2 1 0.015658 34 0.23.415.85.599.962 798.726 d
248.805 560.569 239.028 739.767 3611.526.1492.510.45.92.951
360.815 --undefined-- --undefined--

2 2 0.015158 1157.644 462.472 2517.575 458.882 3806.046 ʒ
525.169 469.444 342.823 --undefined-- --undefined--

2 3 0.015658 2.740.408.5 745.883 454.511 1837.117.675.461 2973.018 e
997.134 4323.468 1710.115 4334

2 4 0.015658 816.110 620.145 180.539 292.343 689.983 ɪ
264.712 1249.624 3264.731 5332.196 3900.344
3.826 --undefined-- --undefined--

2 5 0.015908 860.712 600.877.155.130 219.420 795.560 238.842 971.834 390. dʒ
969 6.685 3937.404 3141.202 --undefined-- --undefined--

2 6 0.015158 3.860.852.455.999.1119.140 eɪ
1623.826 1817.264 2611.081 1095.131
dʒ,ɔɪ,dʒɔɪ
3899.986 19.126 --undefined-- --undefined--

2 7 0.015908 771.032 501.762.100.451.180.201 932.829 2853.413 dʒeɪ
1019.960 3897.537 70.016 5279.085
915.470 --undefined-- --undefined--

2 8 0.015158 740.227 441.458 164.942 ə
1950.594 1554.096 2769.134 961.834 3907.072 269.957
4955.140 660.779 --undefined-- --undefined--

2 9 0.015158 149.774.11.10.064.2645.361 828.519 3950.424 ʊ
566.167 4095.392 2229.650 --undefined-- --undefined--

2 10 0.015158 437.2.752.98.766.612.479.676 əʊ
1714.128 766.891 2480.630 694.977 3751.124
1172.341 4223.515 812.696 --undefined--
417.10.125 546.740 2428.081 922.448 3424.338--undefined--

2 11 0.015908 440.277.165.8.856.686.27.96.717 ʌɪ
545.844 559.428 2423.141 804.721 3478.675

440.347 4546.887 671.767 --undefined-- --undefined--
2 12 0.015158 593.457 4141.752.818.898.35.04.623
1712.922 1202.650 2330.506 1208.750
3362.660 813.465 4379.557 791.455 --undefined-- --undefined-- d
2 13 0.015158 --undefined--22 11 90.408
1685.437 1011.488 2219.072 525.131 445
2.554 603.340 --undefined-- --undefined-- ʒ
2 14 0.015408 834.977 1471.498 2160.205
806.922 4375.087 591.103 --undefined--
2 15 0.015658 43.76.5408.5.816.110.620.145 45.747 4518.992
1137.670 606.547 2358.644 538.861 3582.841
3 299.088 --undefined-- --undefined--
2 16 0.015278 1140.252 436.966 2463.038 501.777 3550.782
442.100 4525.825 491.961 --undefined-- --undefined-- ə
2 17 0.015158 918.198 1066.684 2183.721 890.860 3219.999 1132.889
4387.306 482.831 --undefined-- --undefined--
2 18 0.015008 206.102 932.891 2259.213 788.690 4.
357.912 432.815 --undefined-- --undefined-- ʊ
2 19 0.015287 1125.832.817.947.228.904.928.041.338.269