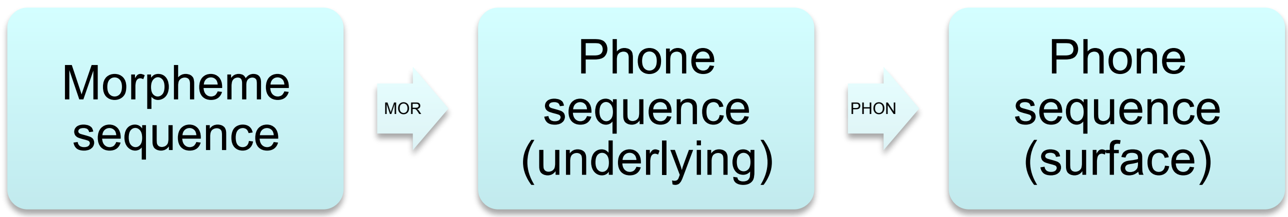
Finite State Speech Recognition

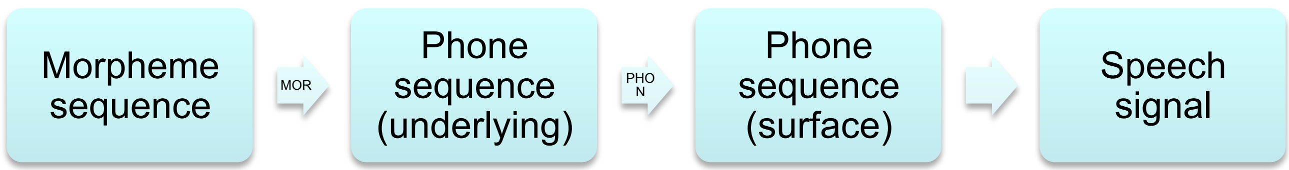
Computational Linguistics

Spring 2014

Finite state pipeline



Finite state pipeline



Main ideas

Weighted finite state machines and weighted transductions.

Vector representation of speech signal: each 10 ms snippet of sound is represented by a vector of floats.

Multivariate gaussian distributions characterize the sounds (vectors) that can be emitted from a state.

```
xfst[0]: regex a+;
688 bytes. 2 states, 2 arcs, Circular.
xfst[1]: print net
Sigma: a
Size: 1.
Flags: deterministic, pruned, minimized, epsilon_free
Arity: 1
s0: a -> fs1.
fs1: a -> fs1.
```

```
0 1 a 1.0
1 1 a 0.5
1 2 0 0.5
2
```

Machines with negative log weights

```
0 1 a 0.0
1 1 a 0.693147182
1 2 0 0.693147182
2
```

Instead of multiplying probabilites, add negative log probabilities.

w(aa) = ?

```
>>> import math
>>> -math.log(0.5)
0.6931471805599453
```

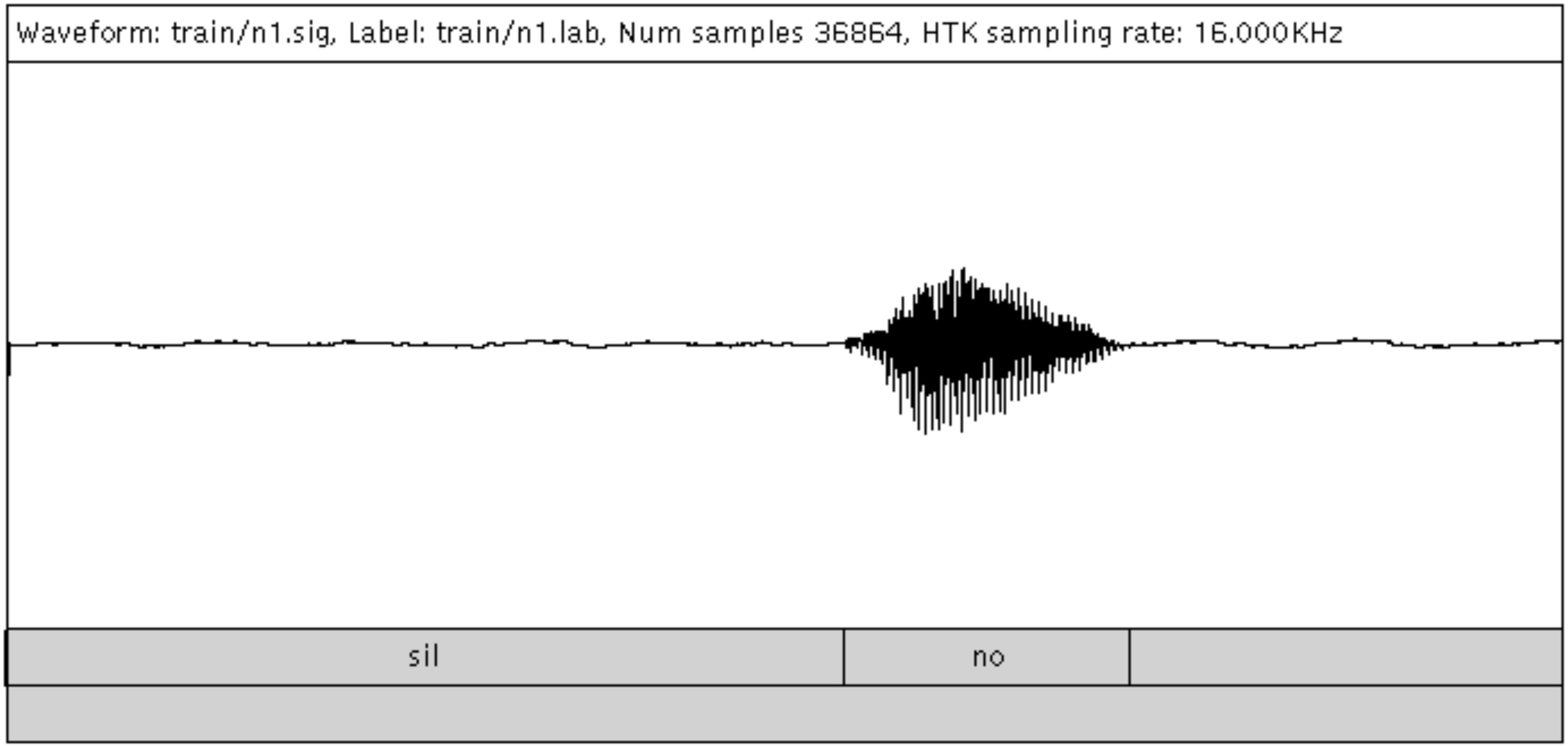
Openfst command line programs

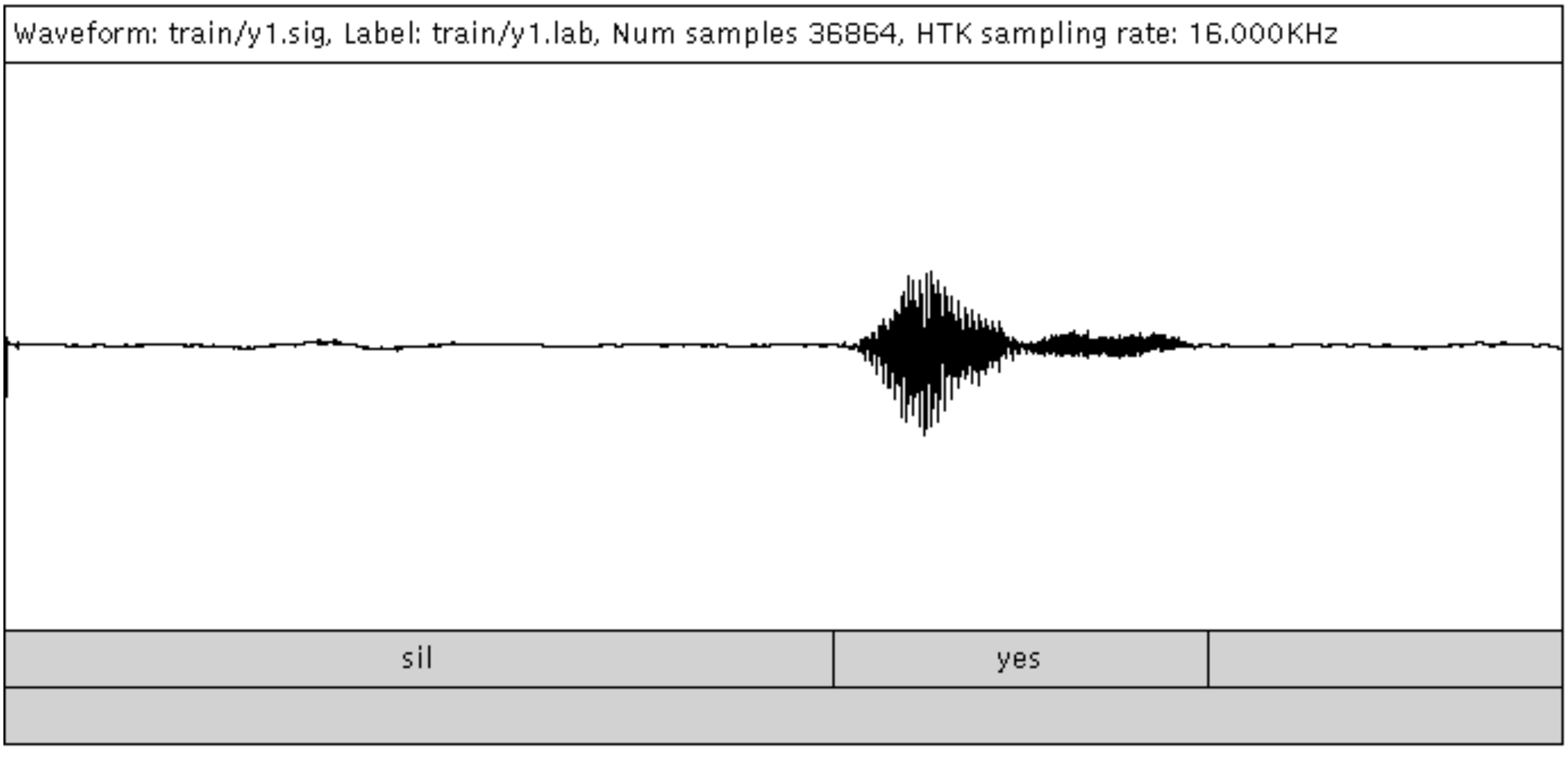
```
m203a:Openfst Mats$ fstcompile --acceptor --isymbols=a.sym b1.txm > b1.fsm
m203a:Openfst Mats$ fstprint --isymbols=a.sym b1.fsm
0       1       a       1       0.693147182
0       1       b       2       0.693147182
1
m203a:Openfst Mats$ fstconcat b1.fsm b1.fsm b1b.fsm
m203a:Openfst Mats$ fstprint --isymbols=a.sym b1b.fsm
0       1       a       1       0.693147182
0       1       b       2       0.693147182
1       2       -       0
2       3       a       1       0.693147182
2       3       b       2       0.693147182
3
```

Yesno problem

Two “sentences”
sil yes sil
sil no sil

(play some)


Waveform: train/n1.sig, Label: train/n1.lab, Num samples 36864, HTK sampling rate: 16.000KHz
sil
no


Waveform: train/y1.sig, Label: train/y1.lab, Num samples 36864, HTK sampling rate: 16.000KHz
sil
yes

Signal file is a sequence of numbers

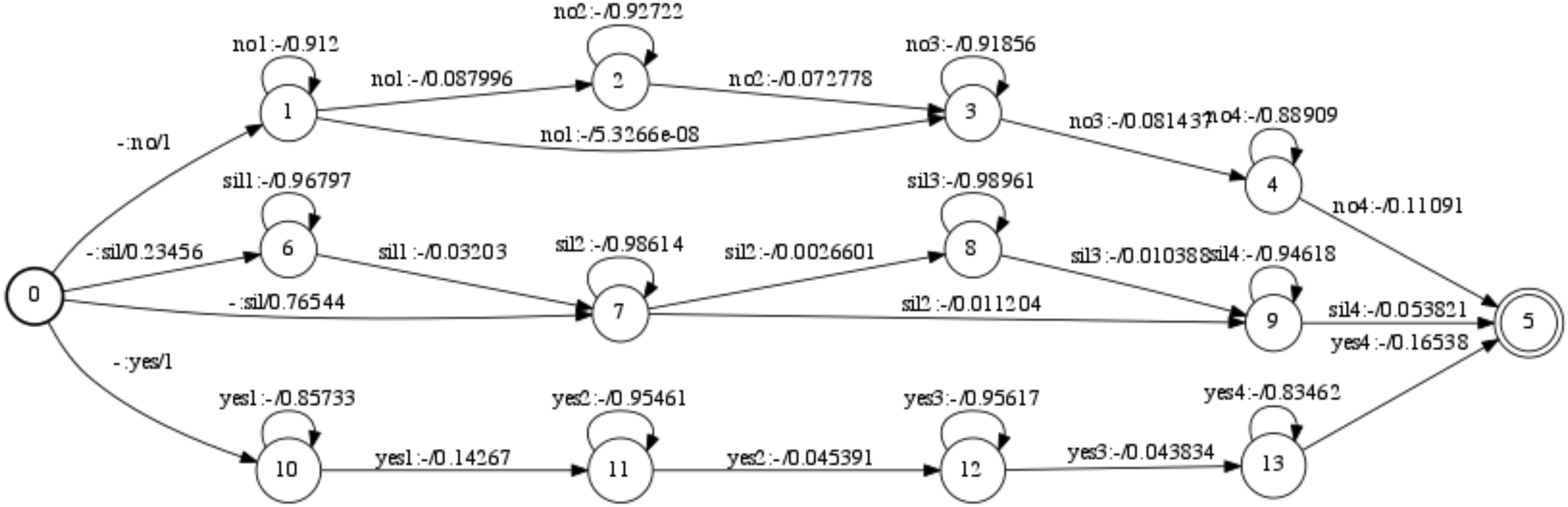
```
vpn16-037:Htk Mats$ HList train/y1.sig | head
----------------------------- Samples: 0->36863 -----------------------------
    0:       0       0       0   -3388   -3228   -3227   -3227   -3226   -3225   -3226
   10:   -3221   -3230   -3211   -3241   -3191   -3267   -3150   -3323   -3066   -3451
   20:   -2831   -4079   -5836     999    -257     310     -57     186      32     134
   30:      56      96      79      93      91      83     102      83      85      85
   40:     101     101     105     106     102      90      99     106     106     112
   50:     108     120     110     120     101     111      95     104     105     109
   60:     114     116     101     109      91      95     106     110     118     123
   70:     112     123     118      99     106     111     116     103     124     102
   80:     117     107      97     104     101     101     103     107      90     105
```

It is converted to a vector representation with lower time resolution, but multiple dimensions.

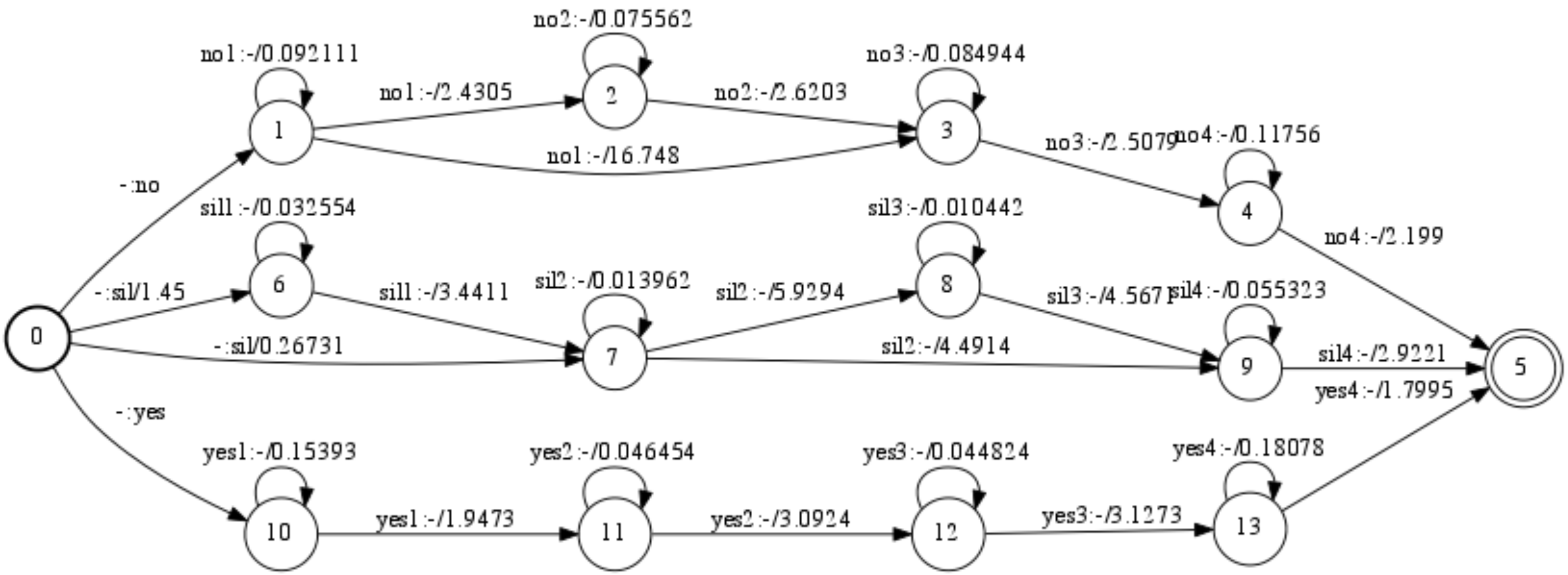
```
vpn16-037:Htk Mats$ HList train/y1.mfcc | head
------------------------------------ Samples: 0
0:      -14.270   2.182
1:      -20.992   1.539
2:      -12.020   0.738
3:       -6.493  -0.531
4:       -8.036   0.156
5:      -11.812   1.904
6:      -13.829  -2.110
7:      -11.427   2.286
8:      -12.801   1.325
vpn16-037:Htk Mats$
```

First two mel frequency cepstral coefficients, with 100 frames/second.

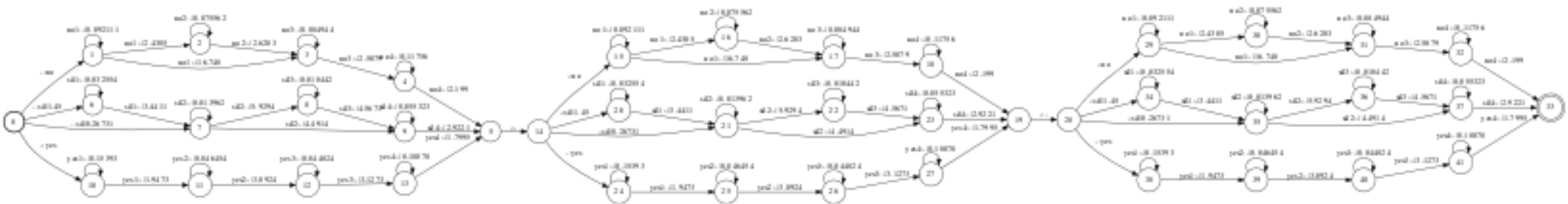
Vocabulary represented by state machine



With log weights

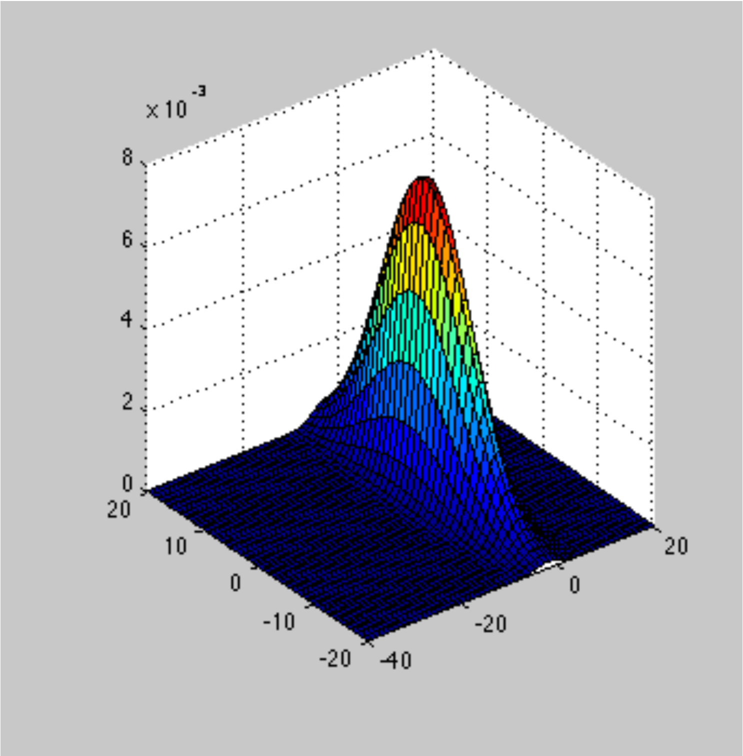


Three words

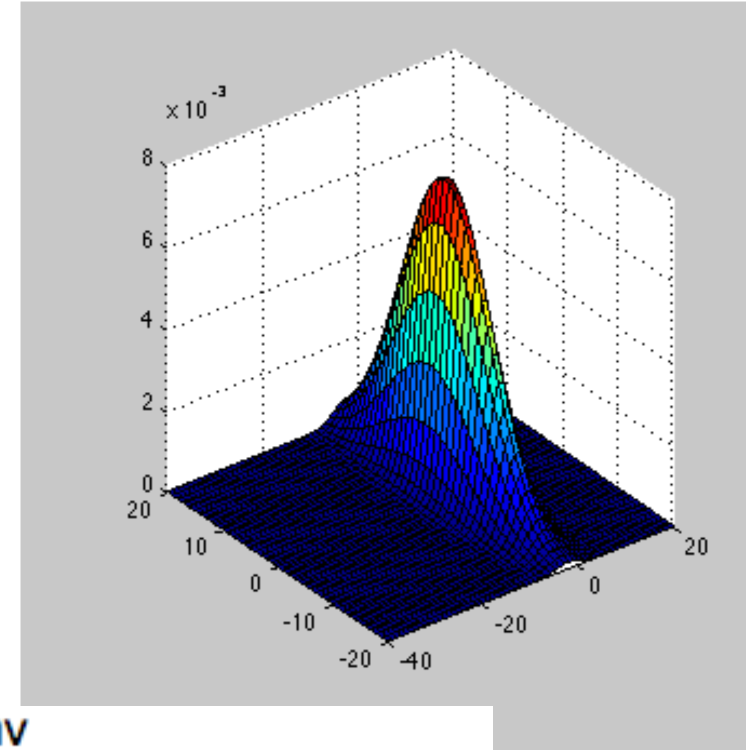


Emitting vectors

States (here no1) come with a multivariate gaussian distribution which can produce any vector, but with different probabilities.

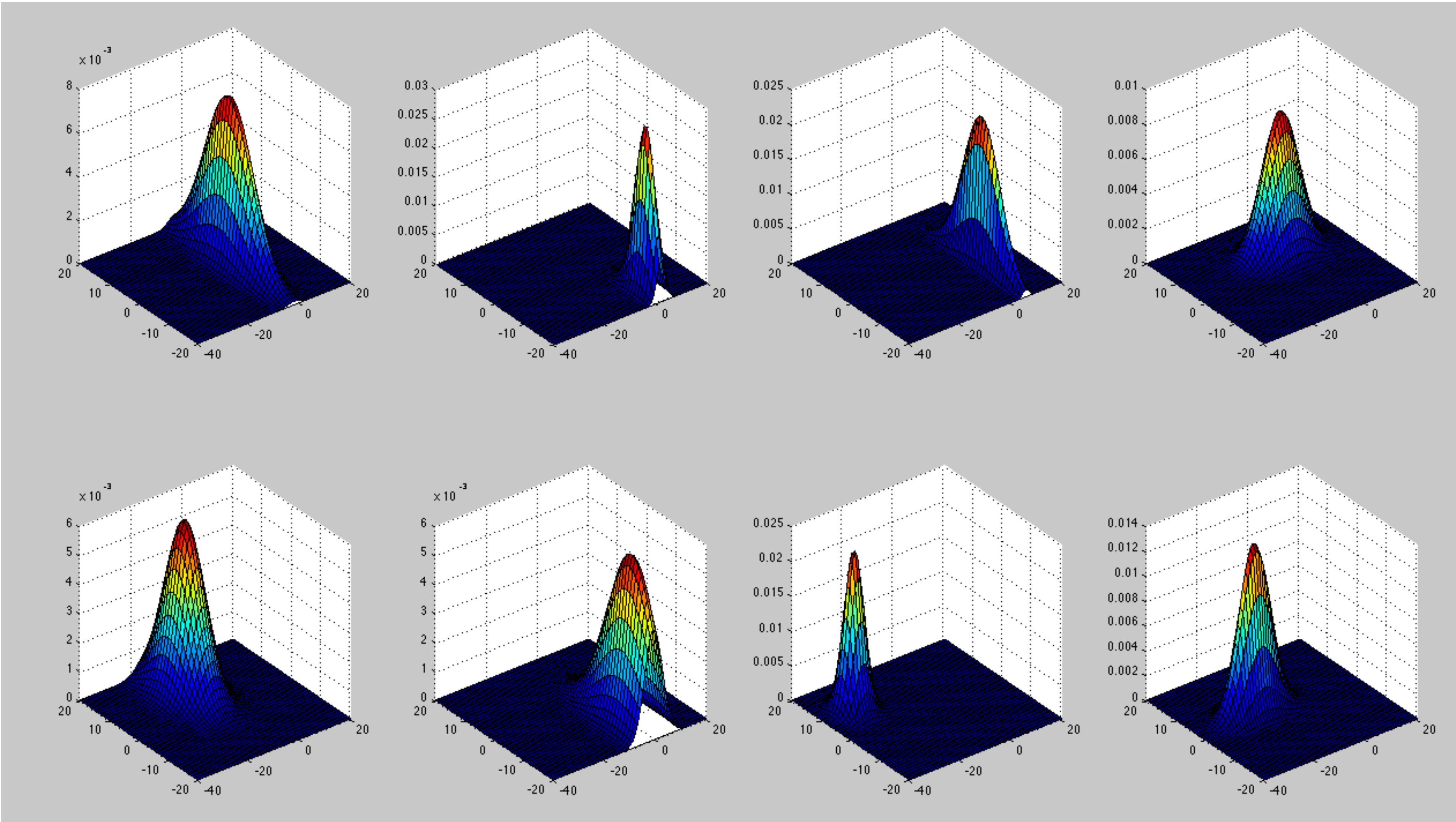


The probability distribution is determined means and standard deviations.



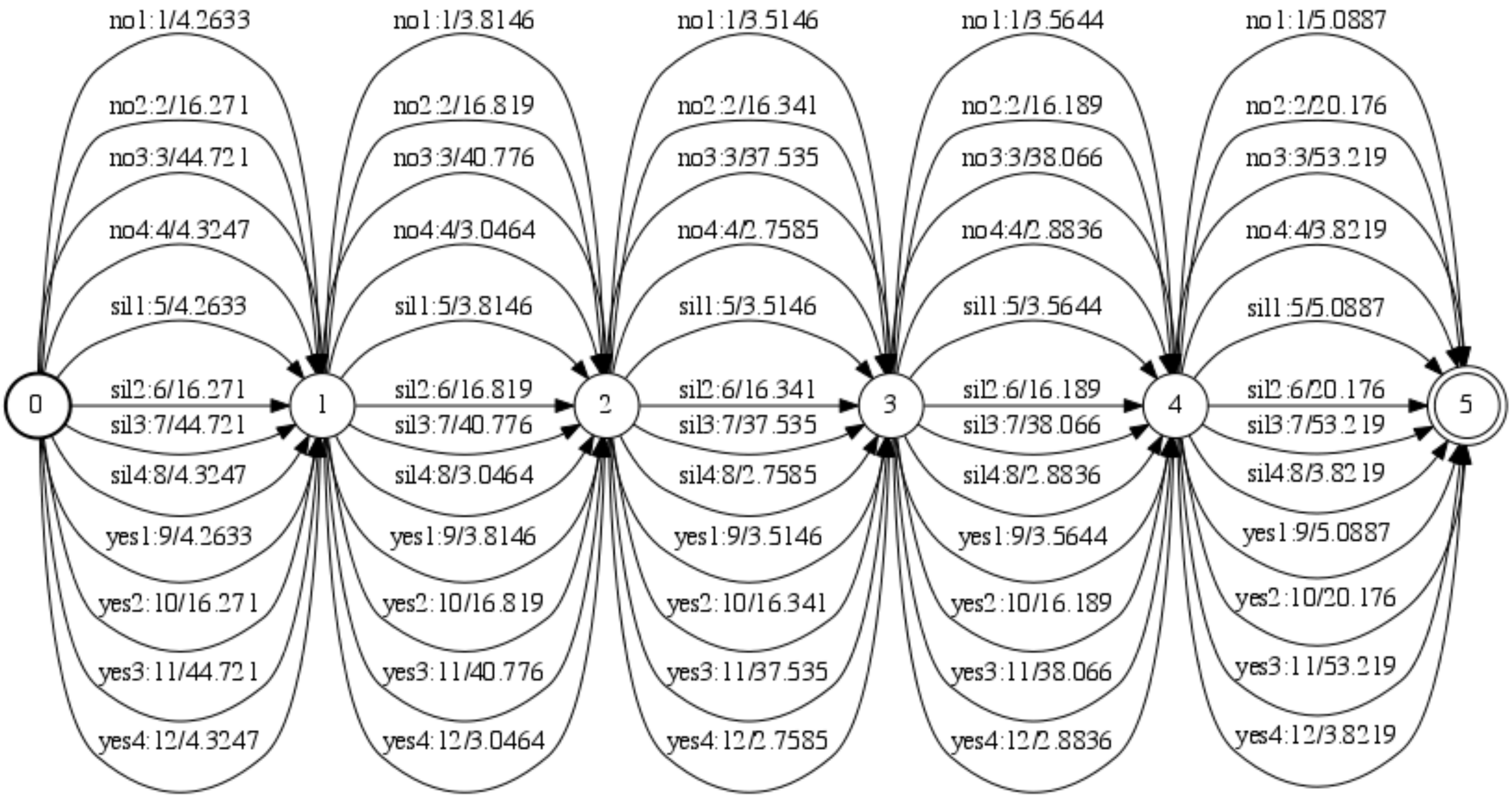

```
vpn16-037:Htk Mats$ cat no_sil_yes.mv
-2.494384e+00 1.796022e+00 6.922693e+00 6.179723e+01
1.830361e+00 -1.457754e+01 2.908821e+00 1.061799e+01
5.765841e+00 -4.699233e+00 1.361376e+00 3.917994e+01
-2.925056e+00 6.590788e+00 4.293019e+01 8.868358e+00
-1.289080e+01 3.939643e+00 1.331727e+01 1.262927e+01
-1.043652e+01 1.780134e+00 1.048423e+00 2.200132e+00
-9.054706e+00 -2.506213e-01 2.046648e+00 4.475942e+00
-1.216040e+01 2.981434e+00 2.005702e+01 1.229846e+01
-1.135676e+01 9.147605e+00 2.139829e+01 3.307150e+01
-7.338192e-01 -1.266852e+01 1.183366e+01 6.107550e+01
-2.786463e+01 9.368406e+00 9.389967e+00 5.049101e+00
-1.768760e+01 2.672389e+00 3.331301e+01 4.034673e+00
```

no1 no2 no3 no4

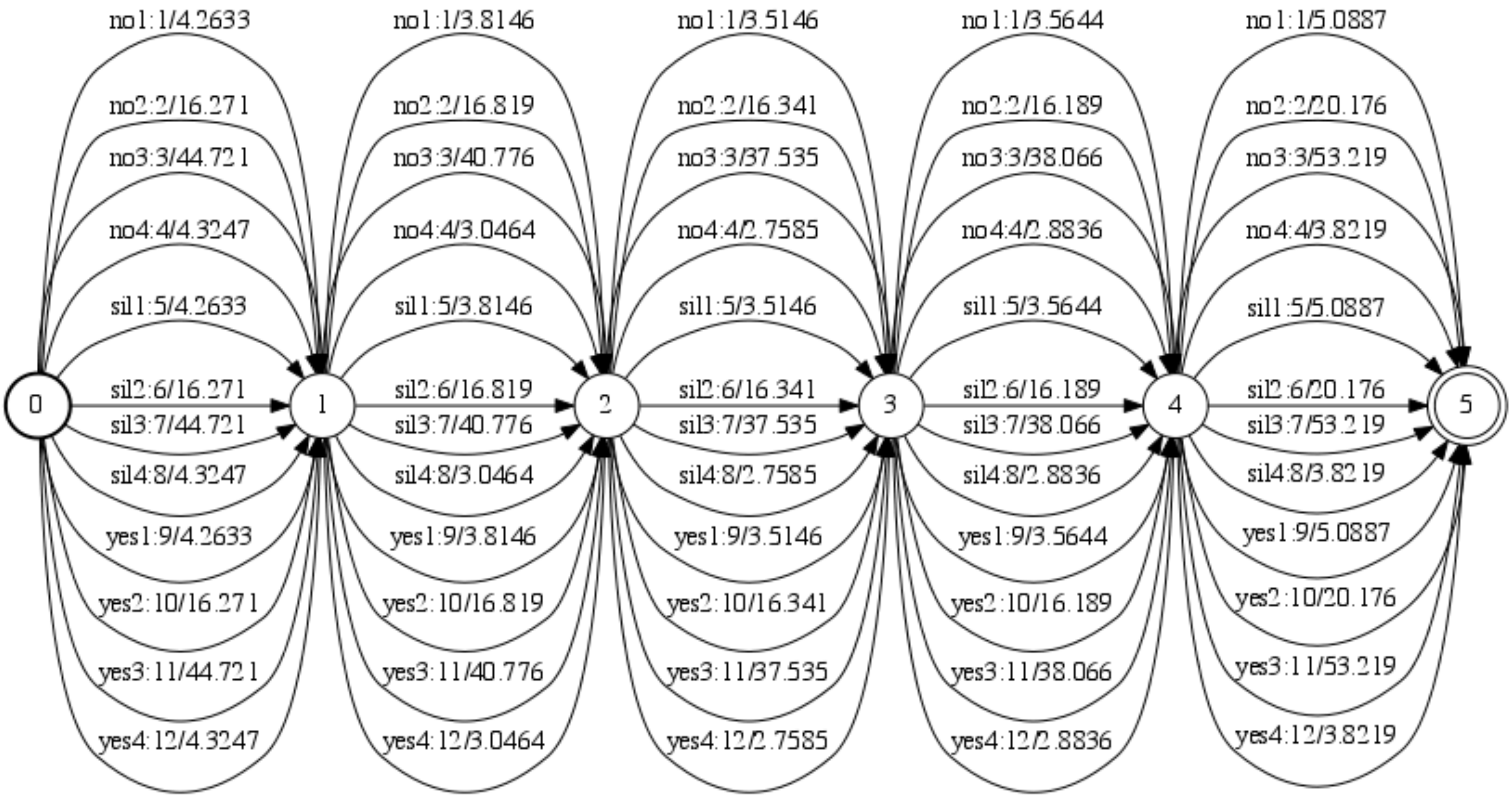


yes1 yes2 yes3 yes4

In each temporal frame, find the cost (negative log probability) of emitting the vector for that frame from each of the twelve states. Assemble the costs into a weighted fsm labeled with word parts.



This machine should be parsed (transduced) into one of the three sentences, by preferring a parse with minimal const.

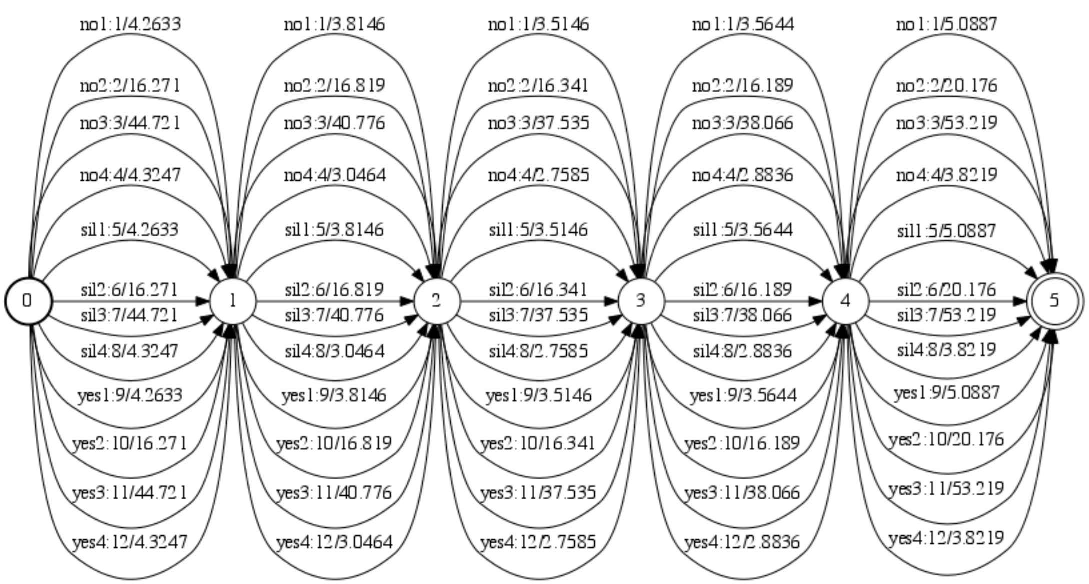


```
m203a:Openfst Mats$ make test/y10.out
cat test/y10.cst | gawk -f Awk/cst2fst.awk  > test/y10.txt1
fstcompile --isymbols=yesno.sym --acceptor test/y10.txt1 > test/y10.fsm1
fstcompose test/y10.fsm1 word3.fst test/y10.fst2
fstcompose test/y10.fst2 utt.fsm test/y10.fst3
fstproject test/y10.fst3 test/y10.fsm4
fstshortestpath test/y10.fsm4 test/y10.fsm5
fsttopsort test/y10.fsm5 test/y10.fsm6
fstprint --isymbols=yesno.sym test/y10.fsm6 > test/y10.out
```

```
m203a:Openfst Mats$ make test/y10.out
cat test/y10.cst | gawk -f Awk/cst2fst.awk  > test/y10.txt1
fstcompile --isymbols=yesno.sym --acceptor test/y10.txt1 > test/y10.fsm1
```

Compile machine representing costs for word-parts in each frame.

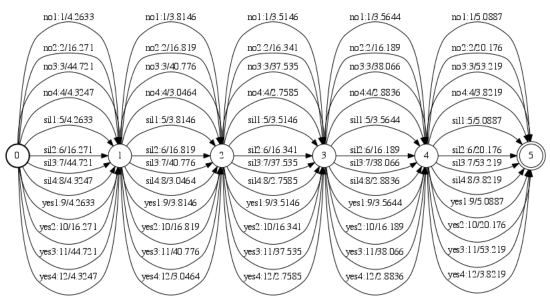
Set of weighted strings, in the vocabulary no1, yes1, sil1, ...



```
fstcompose test/y10.fsm1 word3.fst test/y10.fst2
```

Compose SOU with the the machine WORD3 that represents all three-word sequences. WORD3 has word-parts on the upper side, and words on the lower side.

SOU .o. WORD3



.o.

```
4792    4795    12    0    2.40078092
4793    4796    8     0    4.9520998
4794    4796    4     0    4.77899837
4795    4796    12    0    4.05949926
4796
m203a:Openfst Mats$ fstprint test/y10.fst2 | wc -l
   10153
```

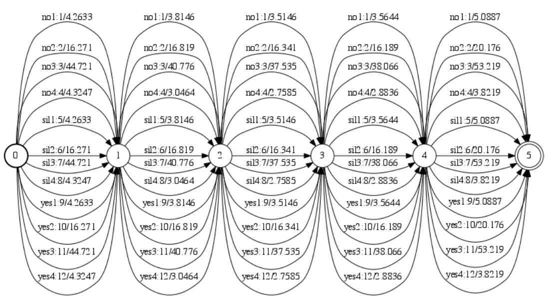
= machine with 4796 states and 10153 arcs.

The number of *paths* is huge.

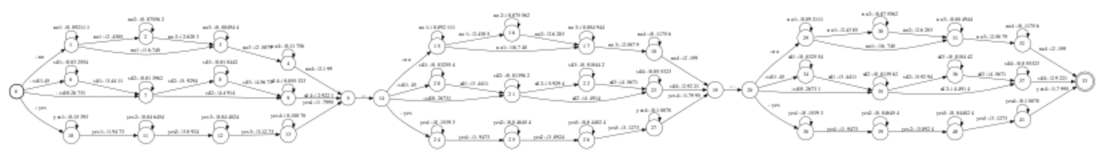
```
fstcompose test/y10.fst2 utt.fsm test/y10.fst3
```

Restrict on the lower side to the set of target sentences UTT.

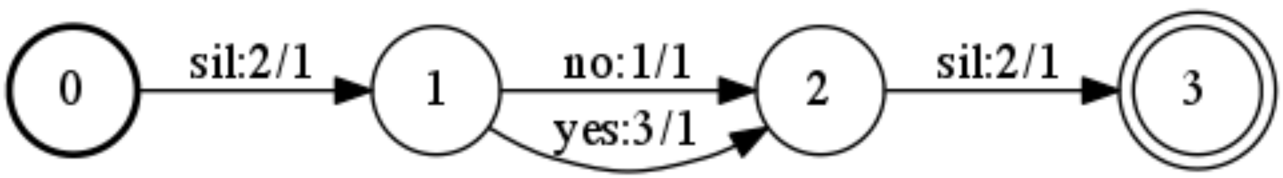
SOU .o. WORD3 .o. UTT



.o.



.o.



```
fstproject test/y10.fst3 test/y10.fsm4
fstshortestpath test/y10.fsm4 test/y10.fsm5
fsttopsort test/y10.fsm5 test/y10.fsm6
fstprint --isymbols=yesno.sym test/y10.fsm6 > test/y10.out
```

Find the path with minimum cost and print it.

61	62	sil4	8	2.15532279
62	63	sil4	8	2.55532289
63	64	sil4	8	5.52209997
64	65	-	0	
65	66	-	0	1
66	67	yes1	9	3.3539269
67	68	yes1	9	3.57392693
68	69	yes1	9	2.97392678
69	70	yes1	9	2.43392682
70	71	yes1	9	3.07392693
71	72	yes1	9	3.33392692
72	73	yes1	9	3.6039269
73	74	yes1	9	4.86725426
74	75	yes2	10	3.2064538
75	76	yes2	10	2.43645382

```
m203a:Openfst Mats$ head -107 test/n10.out | tail
97      98      sil4    8       4.35532331
98      99      sil4    8       2.84532285
99      100     sil4    8       3.36532283
100     101     sil4    8       5.07210016
101     102     -       0
102     103     -       0       1
103     104     no1     1       2.24211121
104     105     no1     1       2.33211112
105     106     no1     1       2.32211113
106     107     no1     1       2.65211105
m203a:Openfst Mats$
```

Why were these a help?

Weighted finite state machines and weighted transductions.

Vector representation of speech signal: each 10 ms snippet of sound is represented by a vector of floats.

Multivariate gaussian distributions characterize the sounds (vectors) that can be emitted from a state.

HTK (Entropic/Microsoft) Compute MFCC, estimate acoustic and word models.

Matlab Evaluate gaussian distributions.

Openfst (Google) Computations with weighted transducers.

Python+Awk Glue.