Finite-State
and the Noisy Channel
Word Segmentation

\[ x = \textit{theprophetsaidtothecity} \]

- What does this say?
  - And what other words are substrings?
- Could segment with parsing (how?), but slow.
- Given \( L \) = a “lexicon” FSA that matches all English words.
- What do these regexps do?
  - \( L^* \)
  - \( L^* \& x \) (what does this FSA look like? how many paths?)
  - \( [ L \ " \ ]^* \)
  - \( [ L \ " \ " : 0 ]^* . o . x \)
  - \( [ L \ " \ ]^* . o . [ [ " \ " : 0 ] | \ " \ ] \]* . o . x \)

(equivalent)
Word Segmentation

\[ x = \text{theprophetsaidtothecity} \]

- What does this say?
  - And what other words are substrings?
- Could segment with parsing (how?), but slow.

- Given \( L \) = a “lexicon” FSA that matches all English words.
- Let’s try to recover the spaces with a single regexp ...

\[
[ \ [ L \ \"\" : 0 \] \] \ . o . \ x \ ] . u
\]

- word sequence transduced to spaceless version
- restrict to paths that produce observed string \( x \)
- language of upper strings of those paths
**Word Segmentation**

\[
x = \text{theprophetsaidtothecity}
\]

- Given \( L \) = a “lexicon” FSA that matches all English words.

- What if Lexicon is weighted?
- From unigrams to bigrams?
- Smooth L to include unseen words?
- Make upper strings be over “alphabet” of words, not letters?

Point of the lecture: This solution generalizes!
Spelling correction

- Spelling correction also needs a lexicon $L$
- But there is distortion beyond deleting spaces ...
  - Let $T$ be a transducer that models common typos and other spelling errors
    - $\text{ance} \rightarrow \text{ence}$ (deliverance, ...)
    - $e \rightarrow \varepsilon$ (deliverance, ...)
    - $\varepsilon \rightarrow e // \text{Cons } _{\text{Cons}}$ (athlete, ...)
    - $rr \rightarrow r$ (embarrass occurrence, ...)
    - $ge \rightarrow dge$ (privilege, ...)
    - etc.
  - What does $L . o . T$ represent? How’s that useful?
- Should $L$ and $T$ have probabilities?
- We’ll want $T$ to include “all possible” errors ...
Noisy Channel Model

real language $Y$

noisy channel $Y \rightarrow X$

observed string $X$

want to recover $Y$ from $X$
Noisy Channel Model

real language \( Y \)

noisy channel \( Y \rightarrow X \)

observed string \( X \)

want to recover \( Y \) from \( X \)

- correct spelling
- typos
- misspelling
Noisy Channel Model

real language $Y$

noisy channel $Y \rightarrow X$

observed string $X$

(lexicon space)*
delete spaces
text w/o spaces

want to recover $Y$ from $X$
Noisy Channel Model

real language \( Y \) → (lexicon space)*

noisy channel \( Y \rightarrow X \) pronunciation

observed string \( X \) speech

want to recover \( Y \) from \( X \)
Noisy Channel Model

real language $Y$

probabilistic CFG

noisy channel $Y \rightarrow X$

delete everything but terminals

tree

observed string $X$

text

want to recover $Y$ from $X$
Noisy Channel Model

real language $Y$

noisy channel $Y \rightarrow X$

observed string $X$

$p(Y)$

$p(X | Y)$

$\ast$

$= p(X, Y)$

want to recover $y \in Y$ from $x \in X$

choose $y$ that maximizes $p(y | x)$ or equivalently $p(x, y)$
Remember:
Noisy Channel and Bayes’ Theorem

```
Remember:
Noisy Channel and Bayes’ Theorem

```

![Diagram](image)

```
Remember:
Noisy Channel and Bayes’ Theorem

In a “noisy channel,” y is transformed into x.

\[ p(Y=y) \Rightarrow p(X=x | Y=y) \]

The “decoder” aims to find the most likely reconstruction of y.

\[ \text{maximize } p(Y=y | X=x) \]

\[ = p(Y=y) p(X=x | Y=y) / \sum_{y'} p(Y=y') p(X=x | Y=y') \]

600.465 - Intro to NLP - J. Eisner
Noisy Channel Model

\[ p(Y) \]

\[ p(X \mid Y) \]

\[ p(X, Y) \]

Note \( p(x, y) \) sums to 1.

Suppose \( x=\text{“C”}; \) what is best “y”?
Noisy Channel Model

Suppose \( x = "C" \); what is best \( "y" \)?

\[
p(Y) * \quad p(X \mid Y) = \quad p(X,Y)
\]
Noisy Channel Model

\[ p(Y) \]

\[ p(X \mid Y) \]

\[(X==x)? \quad 0 \text{ or } 1\]

\[p(x,Y)\]

Restrict just to paths compatible with output “C”
Noisy Channel Model

(i.e., paths transducing $y \rightarrow x$
have total prob $p(x \mid y)$)

noisy channel FST:
accepts pair $(x, y)$
with weight $p(x \mid y)$

source model FSA:
accepts string $y$
with weight $p(y)$

(i.e., paths accepting $y$
have total prob $p(y)$)

restrict to paths
that produce
observed string $x$

language of
upper strings
of those paths
Just Bayes’ Theorem Again

\[ p(Y) = \text{prior on } Y \]
\[ p(X \mid Y) = \text{likelihood of } Y, \text{ for each } X \]
\[ \text{evidence } X=x \]
\[ \begin{align*}
\text{p(x,Y): divide by constant } p(x) \text{ to get posterior } p(Y \mid x).
\end{align*} \]
Morpheme Segmentation

- Let Lexicon be a machine that matches all Turkish words
  - Same problem as word segmentation
  - Just at a lower level: morpheme segmentation
  - Turkish word: uygarlaştıramadıklarımızdanmışsınızcasına
    = uygar+laş+tır+ma+dık+ları+mız+dan+mış+sınız+ca+sı+na
    (behaving) as if you are among those whom we could not cause to become civilized
  - Some constraints on morpheme sequence: bigram probs
  - Generative model – concatenate then fix up joints
    - stop + -ing = stopping, fly + -s = flies, vowel harmony
    - Use a cascade of transducers to handle all the fixups
  - But this is just morphology!
  - Can use probabilities here too (but people often don’t)
An FST: Baby Think & Baby Talk

**FST mapping think to talk**

(☹️ = just babbling, no thought)

---

observe
talk

---

recover
think, by composition

---

Mama/.005
Mama Iwant/.0005
Mama Iwant Iwant/.00005

---

600.465 - Intro to NLP - J. Eisner
Joint Prob. by Double Composition

\[ p(\Sigma^* : mm) = .0375555 = \text{sum of all paths} \]
Cute method for summing all paths

\[ p(\Sigma^* : mm) = 0.0375555 = \text{sum of all paths} \]
Cute method for summing all paths

\[ \text{think} \]

\[ \varepsilon : \Sigma \]

\[ \varepsilon : m / 0.05 \]

\[ \text{IWant} : u / 0.8 \]

\[ \text{Mama} : m \]

\[ \text{IWant} : \varepsilon / 0.1 \]

\[ \text{talk} \]

\[ m : \varepsilon \]

\[ m : \varepsilon \]

\[ m : \varepsilon \]

\[ \text{compose} \]

\[ \varepsilon : \varepsilon / 0.05 \]

\[ \varepsilon : \varepsilon \]

\[ \varepsilon : \varepsilon / 0.1 \]

\[ \varepsilon : \varepsilon / 0.4 \]

\[ \varepsilon : \varepsilon / 0.4 \]

\[ \text{p} (\Sigma^* : \text{mm}) = 0.0375555 = \text{sum of all paths} \]
Cute method for summing all paths

think

```
\epsilon: \Sigma
```

talk

```
\epsilon: m / .05
```

```
\epsilon: b / .2
```

```
\epsilon: m / .4
```

```
IWant: u / .8
```

```
Mama: m
```

```
IWant: \epsilon / .1
```

compose + minimize

```
0.0375555
```

```
p(\Sigma^* : mm) = .0375555 = \text{sum of all paths}
```
p(Mama ... | mm)

\[
p(Mama \Sigma^* | mm) = \frac{.005555}{.0375555} = .148
\]

\[
p(Mama \Sigma^* : mm) = .005555 = \text{sum of Mama ... paths}
\]

\[
p(\Sigma^* : mm) = .0375555 = \text{sum of all paths}
\]
Edit Distance

- Older baby said *caca*
- Was probably thinking *clara* (?)

Do these match up well? How well?

<table>
<thead>
<tr>
<th>clara</th>
<th>clar a</th>
<th>clara</th>
<th>clara</th>
</tr>
</thead>
<tbody>
<tr>
<td>caca</td>
<td>caca</td>
<td>caca</td>
<td>caca</td>
</tr>
</tbody>
</table>

- 3 substitutions + 1 deletion = total cost 4
- 2 deletions + 1 insertion = total cost 3
- 1 deletion + 1 substitution = total cost 2
- 5 deletions + 4 insertions = total cost 9

**minimum edit distance (best alignment)**
Edit distance as min-cost path

Minimum-cost path shows the best alignment, and its cost is the edit distance.
Edit distance as min-cost path

Minimum-cost path shows the best alignment, and its cost is the edit distance.
**Edit distance as min-cost path**

A deletion edge has cost 1.

It advances in the upper string only, so it's horizontal.

It pairs the next letter of the upper string with $\varepsilon$ (empty) in the lower string.
An **insertion edge** has cost 1.

It advances in the lower string only, so it’s vertical.

It pairs $\varepsilon$ (empty) in the upper string with the next letter of the lower string.
Edit distance as min-cost path

A substitution edge has cost 0 or 1.

It advances in the upper and lower strings simultaneously, so it’s diagonal.

It pairs the next letter of the upper string with the next letter of the lower string.

Cost is 0 or 1 depending on whether those letters are identical!
Edit distance as min-cost path

We're looking for a path from upper left to lower right (so as to get through both strings)

Solid edges have cost 0, dashed edges have cost 1

So we want the path with the fewest dashed edges
Edit distance as min-cost path

clara  
caca  
3 substitutions  
+ 1 deletion  
= total cost 4
Edit distance as min-cost path

clar a
c a ca

2 deletions
+ 1 insertion
= total cost 3
Edit distance as min-cost path

clara
c aca

1 deletion
+ 1 substitution
= total cost 2
Edit distance as min-cost path

clara
caca

5 deletions
+ 4 insertions
= total cost 9
Edit Distance Transducer

- **O(k)** deletion arcs
- **O(k^2)** substitution arcs
- **O(k)** insertion arcs
- **O(k)** no-change arcs
Stochastic

Edit Distance Transducer

\(O(k)\) deletion arcs

\(O(k^2)\) substitution arcs

\(O(k)\) insertion arcs

\(O(k)\) identity arcs

Likely edits = high-probability arcs

Defines \(p(x \mid y)\), e.g., \(p(caca \mid clara)\)
Most likely path that maps clara to caca?

![Diagram showing the best path (by Dijkstra’s algorithm)]

Best path (by Dijkstra’s algorithm)
What string produced \textit{caca}? (posterior distribution)

Lexicon*

\begin{itemize}
  \item \textit{caca}
  \item $\varepsilon : a$
  \item $\varepsilon : b$
  \item $a : \varepsilon$
  \item $b : \varepsilon$
  \item $a : a$
  \item $b : a$
  \item $a : b$
  \item $b : b$
  \item $\varepsilon : \varepsilon$
\end{itemize}

\begin{itemize}
  \item more complicated FST, with cycles.
  \item Each path is a possible input string aligned to \textit{caca}.
  \item A path’s weight is proportional to the posterior probability of that path as an explanation for \textit{caca}. Could find best path.
\end{itemize}
Speech Recognition by FST Composition (Pereira & Riley 1996)

- **trigram language model**: \( p(\text{word seq}) \)
- **pronunciation model**: \( p(\text{phone seq} \mid \text{word seq}) \)
- **acoustic model**: \( p(\text{acoustics} \mid \text{phone seq}) \)
- **observed acoustics**
trigram language model

\[ p(\text{word seq}) \]

\[ p(\text{phone seq} | \text{word seq}) \]

\[ p(\text{acoustics} | \text{phone seq}) \]
HMM Tagging

- Can think of this as a finite-state noisy channel, too!
- See slides in the “tagging” lecture.