# CSCI 5832 Natural Language Processing

Lecture 5
Jim Martin

## Today 1/29

- Review
- FSTs/Composing cascades
- Break
- FST Examples
  - Porter Stemmer
  - Soundex
- Segmentation
- Edit distance

## FST Review

- FSTs allow us to take an input and deliver a structure based on it
- Or… take a structure and create a surface form
- Or take a structure and create another structure

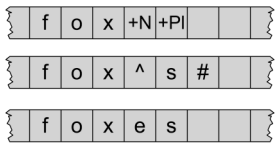
## FST Review

- What does the transition table for an FSA consist of?
- How does that change with an FST?

## Review

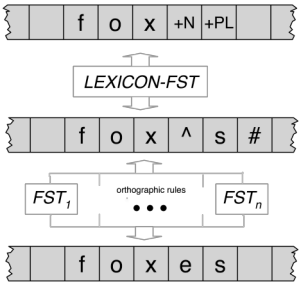
- In many applications it's convenient to decompose the problem into a set of cascaded transducers where
  - The output of one feeds into the input of the next.
  - We'll see this scheme again for deeper semantic processing.

## English Spelling Changes

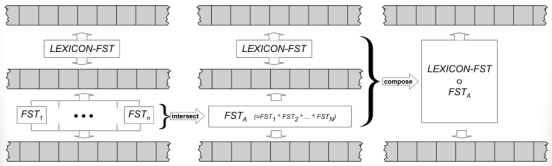



- We use one machine to transduce between the lexical and the intermediate level, and another to handle the spelling changes to the surface tape

## Overall Plan




## Final Scheme




## Composition

1. Create a set of new states that correspond to each pair of states from the original machines (New states are called (x,y), where x is a state from M1, and y is a state from M2)
2. Create a new FST transition table for the new machine according to the following intuition…

## Composition

- There should be a transition between two states in the new machine if it's the case that the output for a transition from a state from M1, is the same as the input to a transition from M2 or…

## Composition

- $\delta_3((x_a,y_a), i:o) = (x_b,y_b)$ iff
  - There exists c such that
  - $\delta_1(x_a, i:c) = x_b$ AND
  - $\delta_2(y_a, c:o) = y_b$

## Light Weight Morphology

- Sometimes you just need to know the stem of a word and you don't care about the structure.
- In fact you may not even care if you get the right stem, as long as you get a consistent string.
- This is stemming… it most often shows up in IR applications

## Stemming for Information Retrieval

- Run a stemmer on the documents to be indexed
- Run a stemmer on users' queries
- Match
  - This is basically a form of hashing, where you want collisions.

## Porter

- No lexicon needed
- Basically a set of staged sets of rewrite rules that strip suffixes
- Handles both inflectional and derivational suffixes
- Doesn't guarantee that the resulting stem is really a stem (see first bullet)
- Lack of guarantee doesn't matter for IR

## Porter Example

- Computerization
  - ization -> -ize computerize
  - ize -> ε computer

## Porter

- The original exposition of the Porter stemmer did not describe it as a transducer but…
  - Each stage is separate transducer
  - The stages can be composed to get one big transducer
  - http://tartarus.org/martin/PorterStemmer/

## Next HW

- Part 1: Alter your code to take newswire text and segment it into paragraphs, sentences and words. As in…

```
<p>
<s> The quick brown fox. </s>
<s> This is sentence two. </s>
</p>
```

## Readings/Quiz

- First quiz is 2/12 (2 weeks from today)
- It will cover Chapters 2,3,4,5 and parts of 6
- Lectures are based on the assumption that you've read the text before class.
- Quizzes are based on the contents of the lectures and the readings.

## Segmentation

- A significant part of the morphological analysis we've been doing involves segmentation
  - Sometimes getting the segments is all we want
    - Morpheme boundaries, words, sentences, paragraphs

## Segmentation

- Segmenting words in running text
- Segmenting sentences in running text
- Why not just periods and white-space?
  - `Mr. Sherwood said reaction to Sea Containers' proposal has been "very positive." In New York Stock Exchange composite trading yesterday, Sea Containers closed at $62.625, up 62.5 cents.`
  - `"I said, 'what're you? Crazy?' " said Sadowsky. "I can't afford to do that.''`
- Words like:
  - `cents.  said,  positive."  Crazy?`

## Can't Just Segment on Punctuation

- Word-internal punctuation
  - M.p.h
  - Ph.D.
  - AT&T
  - 01/02/06
  - Google.com
  - 555,500.50
- Expanding clitics
  - What're -> what are
  - I'm -> I am
- Multi-token words
  - New York
  - Rock 'n' roll

## Sentence Segmentation

- !, ? relatively unambiguous
- Period "." is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
- General idea:
  - Build a binary classifier:
    - Looks at a "."
    - Decides EndOfSentence/NotEOS
    - Could be hand-written rules, or machine-learning

## Word Segmentation in Chinese

- Some languages don't have spaces
  - Chinese, Japanese, Thai, Khmer
- Chinese:
  - Words composed of characters
  - Characters are generally 1 syllable and 1 morpheme.
  - Average word is 2.4 characters long.
  - Standard segmentation algorithm:
    - Maximum Matching (also called Greedy)

## Maximum Matching Word Segmentation

Given a wordlist of Chinese, and a string.

1) Start a pointer at the beginning of the string
2) Find the longest word in dictionary that matches the string starting at pointer
3) Move the pointer over the word in string
4) Go to 2

## English example (Palmer 00)

- the table down there
- thetabledownthere
- Theta bled own there

- Works astonishingly well in Chinese
- Far better than this English example suggests
- Modern algorithms better still:
  - Probabilistic segmentation

## Spelling Correction and Edit Distance

- Non-word error detection:
  - detecting “graffe”
- Non-word error correction:
  - figuring out that “graffe” should be “giraffe”
- Context-dependent error detection and correction:
  - Figuring out that “war and piece” should be peace

## Non-Word Error Detection

- Any word not in a dictionary is a spelling error
- Need a big dictionary!
- What to use?
  - FST dictionary!!

## Isolated Word Error Correction

- How do I fix “graffe”?
  - Search through all words:
    - graf
    - craft
    - grail
    - giraffe
  - Pick the one that’s closest to graffe
  - What does “closest” mean?
  - We need a distance metric.
  - The simplest one: edit distance
    - I.e. Unix diff
    - (More sophisticated probabilistic ones: noisy channel)

## Edit Distance

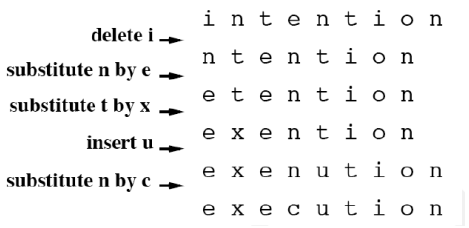
- The minimum edit distance between two strings is the minimum number of editing operations
  - Insertion
  - Deletion
  - Substitution
- Needed to transform one string into the other

## Minimum Edit Distance

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
d s s   i s
```

- If each operation has cost of 1
- Distance between these is 5
- If substitutions cost 2 (Levenshtein)
- Distance between these is 8

## Min Edit Example




## Min Edit As Search

- But that generates a huge search space
  - Initial state is the word we're transforming
  - Operators are insert, delete, substitute
  - Goal state is the word we're trying to get to
  - Path cost is what we're trying to minimize
- Navigating that space in a naïve backtracking fashion would be incredibly wasteful
- Why?

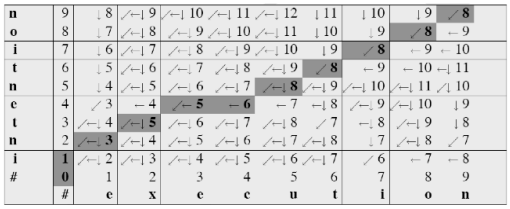
Lots of paths wind up at the same states. But there's no need to keep track of the them all. We only care about the shortest path to each of those revisited states.

## Min Edit Distance

**function** MIN-EDIT-DISTANCE(*target*, *source*) **returns** *min-distance*

$n \leftarrow$ LENGTH(*target*)
$m \leftarrow$ LENGTH(*source*)
Create a distance matrix *distance*[*n*+1,*m*+1]
Initialize the zeroth row and column to be the distance from the empty string
  *distance*[0,0] = 0
  **for** each column *i* **from** 1 **to** *n* **do**
    *distance*[*i*,0] $\leftarrow$ *distance*[*i*-1,0] + *ins-cost*(*target*[*i*])
  **for** each row *j* **from** 1 **to** *m* **do**
    *distance*[0,*j*] $\leftarrow$ *distance*[0,*j*-1] + *del-cost*(*source*[*j*])
**for** each column *i* **from** 1 **to** *n* **do**
  **for** each row *j* **from** 1 **to** *m* **do**
    *distance*[*i*,*j*] $\leftarrow$ MIN( *distance*[*i*−1, *j*] + *ins-cost*(*target*$_{i-1}$),
      *distance*[*i*−1,*j*−1] + *sub-cost*(*source*$_{j-1}$, target$_{i-1}$),
      *distance*[*i*,*j*−1] + *del-cost*(*source*$_{j-1}$))
**return** *distance*[n,m]

## Min Edit Example



| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **n** | 9 | ↓ 8 | ↙←↓ 9 | ↙←↓ 10 | ↙←↓ 11 | ↙←↓ 12 | ↓ 11 | ↓ 10 | ↓ 9 | ↙ **8** |
| **o** | 8 | ↓ 7 | ↙←↓ 8 | ↙←↓ 9 | ↙←↓ 10 | ↙←↓ 11 | ↓ 10 | ↓ 9 | ↙ **8** | ← 9 |
| **i** | 7 | ↓ 6 | ↙←↓ 7 | ↙←↓ 8 | ↙←↓ 9 | ↙←↓ 10 | ↓ 9 | ↙ **8** | ← 9 | ← 10 |
| **t** | 6 | ↓ 5 | ↙←↓ 6 | ↙←↓ 7 | ↙←↓ 8 | ↙←↓ 9 | ↙ **8** | ← 9 | ← 10 | ←↓ 11 |
| **n** | 5 | ↓ 4 | ↙←↓ 5 | ↙←↓ 6 | ↙←↓ 7 | ↙←↓ **8** | ↙←↓ 9 | ↙←↓ 10 | ↙←↓ 11 | ↙↓ 10 |
| **e** | 4 | ↙ 3 | ← 4 | ↙← **5** | ← **6** | ← 7 | ←↓ 8 | ↙←↓ 9 | ↙←↓ 10 | ↓ 9 |
| **t** | 3 | ↙←↓ 4 | ↙←↓ **5** | ↙←↓ 6 | ↙←↓ 7 | ↙←↓ 8 | ↙ 7 | ←↓ 8 | ↙←↓ 9 | ↓ 8 |
| **n** | 2 | ↙←↓ **3** | ↙←↓ 4 | ↙←↓ 5 | ↙←↓ 6 | ↙←↓ 7 | ↙←↓ 8 | ↓ 7 | ↙←↓ 8 | ↙ 7 |
| **i** | **1** | ↙←↓ 2 | ↙←↓ 3 | ↙←↓ 4 | ↙←↓ 5 | ↙←↓ 6 | ↙←↓ 7 | ↙ 6 | ← 7 | ← 8 |
| # | **0** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | # | **e** | **x** | **e** | **c** | **u** | **t** | **i** | **o** | **n** |

## Summary

- Minimum Edit Distance
- A “dynamic programming” algorithm
- We will see a probabilistic version of this called “Viterbi”

## Summary

- Finite State Automata
- Deterministic Recognition of FSAs
- Non-Determinism (NFSAs)
- Recognition of NFSAs
- FSAs, FSTs and Morphology
- Very brief sketch: Tokenization
- Minimum Edit Distance

## Next Time

On to Chapter 4 (N-grams) for Thursday