CS114: Finite State Automata, Words, Transducers

January 22, 2014

Prof. Marie Meteer
Brandeis University

Additional slides courtesy of Jurafsky & Martin, James Pustejovsky and Ray Mooney

Brandeis CS114-2013 Meteer
Assignment 1: Sentence pivots

• Background
  – The theory of “given” and “new” says that the first part of a sentence grounds it in the context (the “given” part) and the second provides information (the “new” part)
  – One study looked at how to find the “pivot” between given and new based on the syntactic structure of the sentence
    • “Modeling Conversational Speech for Speech Recognition” Meteer & Iyer, 1997
  – The goal was to see if the vocabulary and language model for these two parts was different

• Task (part 1)
  – Write a program that uses lexical and part of speech information to split a sentence into its given and new parts
  – Base the split on finding the “first strong verb”
Programming goals

• Get used to Python and NLTK data
• Write a modularized program that separates the declarative rules from the control structure
• Write a program that is meant to be one component in a larger sequence
  – Use internal data structures that can be further modified
  – Separate “read” and “write” functions from the core program since you may not always be writing out the result
  – Put all content specific information in declarative rules so they can be changed for different types of input
Pivot point: After the first strong verb

- **Before the pivot, after the pivot, no pivot**
  - A.1: Uh/UH ,/, do/VBP you/PRP have/VB a/DT pet/NN Randy/NNP ?/.
  - B.2: Uh/UH ,/, yeah/UH ,/, currently/RB we/PRP have/VBP a/DT poodle/NN ./.
  - A.3: A/DT poodle/NN ,/, miniature/JJ or/CC ,/, uh/UH ,/, full/JJ size/NN ?/.
  - B.8: Well/UH ,/, um/UH ,/, I/PRP would/MD n't/RB ,/, uh/UH ,/, I/PRP definitely/RB would/MD n't/RB dispute/VB that/IN
  - B.22: And/CC I/PRP think/VBP ,/, uh/UH ,/, having/VBG listened/VBN to/IN you/PRP relative/JJ to/IN the/DT economy/NN thing/NN
Guidance

• Don’t worry about the theory. Just find the first strong verb
• Follow the programming guidelines
• Keep your rules out of the control structure—you’ll be looking at other kinds of data going forward on the same task
Words

• Finite-state methods are particularly useful in dealing with a lexicon
• Many devices, most with limited memory, need access to large lists of words
• And they need to perform fairly sophisticated tasks with those lists
• So we’ll first talk about some facts about words and then come back to computational methods
Morphology is the study of the ways that words are built up from smaller meaningful units called morphemes.

We can usefully divide morphemes into two classes:

- **Stems**: The core meaning-bearing units
- **Affixes**: Bits and pieces that adhere to stems to change their meanings and grammatical functions
English Morphology

• We can further divide morphology up into two broad classes
  – Inflectional
  – Derivational
Word Classes

• By word class, we have in mind familiar notions like noun and verb
• We’ll go into the gory details in Chapter 5
• Right now we’re concerned with word classes because the way that stems and affixes combine is based to a large degree on the word class of the stem
Inflectional Morphology

• Inflectional morphology concerns the combination of stems and affixes where the resulting word:
  – Has the same word class (PoS) as the original
  – Serves a grammatical/semantic purpose that is
    • Different from the original
    • But is nevertheless transparently related to the original
• Nouns are simple
  – Markers for plural and possessive

• Verbs are only slightly more complex
  – Markers appropriate to the tense of the verb
Regulars and Irregulars

• It is a little complicated by the fact that some words misbehave (refuse to follow the rules)
  – Mouse/mice, goose/geese, ox/oxen
  – Go/went, fly/flew

• The terms regular and irregular are used to refer to words that follow the rules and those that don’t
Regular and Irregular Verbs

- Regulars...
  - Walk, walks, walking, walked, walked

- Irregulars
  - Eat, eats, eating, ate, eaten
  - Catch, catches, catching, caught, caught
  - Cut, cuts, cutting, cut, cut
Verb forms: Not just affixes

- Progressive: be ---ing
- Perfect: have ---ed
- Modality expressed as a word
  - Should, would, could
- Tense affects the first element in the verb group (unless it’s a modal)
FSA for Verb Group Parts of Speech

- **I could have danced** all night: MD VB VBN
- **I was dancing** when the lights went out: VBD VBG
- We **danced** the night away: VBD
- **I would have been dancing**, but ...: MD VB VBN VBG
- He **has danced** his whole life: VBZ VBN
- She **dances** four times a week: VBZ
- He **loves to dance**: VBZ TO VB
- She **might dance** with him: MD VB
Inflectional Morphology

• So inflectional morphology in English is fairly straightforward
• Except that it is highly ambiguous
  – Same endings used for multiple things
    • Plural nouns, present tense 3rd person verbs, possessive
    • Past, perfect, passive
• And complicated by the fact that are irregularities
  – Too many conquerors
Derivational Morphology

- Derivational morphology is the messy stuff that no one ever taught you.
  - Quasi-systematic
  - Irregular meaning change
  - Changes of word class
### Derivational Examples

#### Verbs and Adjectives to Nouns

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Verb</th>
<th>Noun</th>
</tr>
</thead>
<tbody>
<tr>
<td>-ation</td>
<td>computerize</td>
<td>computerization</td>
</tr>
<tr>
<td>-ee</td>
<td>appoint</td>
<td>appointee</td>
</tr>
<tr>
<td>-er</td>
<td>kill</td>
<td>killer</td>
</tr>
<tr>
<td>-ness</td>
<td>fuzzy</td>
<td>fuzziness</td>
</tr>
</tbody>
</table>

#### Nouns and Verbs to Adjectives

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Noun</th>
<th>Adjective</th>
</tr>
</thead>
<tbody>
<tr>
<td>-al</td>
<td>computation</td>
<td>computational</td>
</tr>
<tr>
<td>-able</td>
<td>embrace</td>
<td>embraceable</td>
</tr>
<tr>
<td>-less</td>
<td>clue</td>
<td>clueless</td>
</tr>
</tbody>
</table>
Example: *Compute*

- Many paths are possible...
- Start with *compute*
  - Computer -> computerize -> computerization
  - Computer -> computerize -> computerizable
- But not all paths/operations are equally good (allowable?)
  - Computer -> *Computeree ?? *Computerness??
  - Clue
    - Clue -> *clueable
    - Clueless, Clueful?
    - Unkempt, kempt?, kemptify (meaning to comb one’s hair)
Why care about morphology?

• ‘Stemming’ in information retrieval
  – Might want to search for “going home” and find pages with both “went home” and “will go home”

• Morphology in machine translation
  – Need to know that the Spanish words quiero and quieres are both related to querer ‘want’

• Morphology in spell checking
  – Need to know that misclaim and antiundoggingly are not words despite being made up of word parts
Can’t just list all words

• Turkish

• Uygarlastıramadıklarımızdanmissinizcasina
  – (behaving) as if you are among those whom we could not civilize

• ’Uygar ‘civilized’ + las ‘become’ + tir ‘cause’ + ama ‘not able’ + dik ‘past’ + lar ‘plural’ + imiz ‘p1pl’ + dan ‘abl’ + mis ‘past’ + siniz ‘2pl’ + casina ‘as if’
What we want

• Something to automatically do the following kinds of mappings:
  • Cats    cat +N +PL
  • Cat     cat +N +SG
  • Cities  city +N +PL
  • Merging merge +V +Present-participle
  • Caught  catch +V +past-participle
We’d like to use the machinery provided by FSAs to capture these facts about morphology

– Accept strings that are in the language
– Reject strings that are not
– And do so in a way that doesn’t require us to in effect list all the words in the language
• Regular singular nouns are ok
• Regular plural nouns have an -s on the end
  – Note in speech there are three variants
    • –s, -z, or –ix-z
    • Cats, dogs, bushes
• Irregulars are ok as is
Simple Rules

- **q₀**
  - **reg-noun**
  - **irreg-pl-noun**
  - **irreg-sg-noun**

- **q₁**
  - **plural -s**

- **q₂**
Now Plug in the Words
Derivational Rules

If everything is an accept state how do things ever get rejected?
• We can now run strings through these machines to recognize strings in the language
• But recognition is usually not quite what we need
  – Often if we find some string in the language we might like to assign a structure to it (parsing)
  – Or we might have some structure and we want to produce a surface form for it (production/generation)
• Example
  – From “cats” to “cat +N +PL”
Finite State Transducers

• The simple story
  – Add another tape
  – Add extra symbols to the transitions

  – On one tape we read “cats”, on the other we write “cat +N +PL”
FSTs

**Lexical**

```
cat  +N  +Pl
```

**Surface**

```
cats
```
Applications

• The kind of parsing we’re talking about is normally called **morphological analysis**
• It can either be
  • An important stand-alone component of many applications (spelling correction, information retrieval)
  • Or simply a link in a chain of further linguistic analysis
The Details

- Of course, it's not as easy as
  - “cat +N +PL” <-> “cats”
- As we saw earlier there are geese, mice and oxen
- But there are also a whole host of spelling/pronunciation changes that go along with inflectional changes
  - Cats vs Dogs
  - Fox and Foxes
Multi-Tape Machines

• To deal with these complications, we will add more tapes and use the output of one tape machine as the input to the next
• So to handle irregular spelling changes we’ll add intermediate tapes with intermediate symbols
## Multi-Level Tape Machines

<table>
<thead>
<tr>
<th>Level</th>
<th>Tape</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Lexical</strong></td>
<td>f o x</td>
<td>+N +Pl</td>
</tr>
<tr>
<td><strong>Intermediate</strong></td>
<td>f o x ^ s #</td>
<td></td>
</tr>
<tr>
<td><strong>Surface</strong></td>
<td>f o x e s</td>
<td></td>
</tr>
</tbody>
</table>

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

- The add an “e” rule as in $\text{fox}^s\# \leftrightarrow \text{foxes}\#$

Deterministic or Nondeterministic?
Foxes

Lexical

Intermediate

Surface

$T_{\text{lex}}$

$T_{\text{e-insert}}$
• This is an architecture that we’ll see again and again
  • Overall processing is divided up into distinct rewrite steps
  • The output of one layer serves as the input to the next
  • The intermediate tapes may or may not wind up being useful in their own right
More about Words

• Tokenization: Can’t just take words for granted
  – Finding the words
  – Sentence segmentation
  – Word segmentation

• Spell check and Edit Distance
Tokenization

• Segmenting words and 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.

• Words like: cents. said, positive.” Crazy?
One can’t segment on punctuation alone

• 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

- How about speech recognition?
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
Spell-checking 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
• Assume it’s a spelling error
• Need a big dictionary!
• What to use?
  – FST dictionary!!
    • But what issues did we raise with earlier?
    • Can we use it for all kinds of morphology?
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.
    • (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 into the other
Minimum Edit Distance

IN 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
How to come up with the minimum?

• Try all possibilities

\[
\begin{align*}
\text{INTENTION} & \quad \text{EXECUTION} \\
\text{d} \quad \text{d} \quad \text{d} \quad \text{d} \quad \text{d} \quad \text{d} \quad \text{d} \quad \text{d} & \quad \text{i} \quad \text{i} \quad \text{i} \quad \text{i} \quad \text{i} \quad \text{i} \quad \text{i} \quad \text{i} \quad \text{i} \quad \text{i} \quad \text{i} = 18
\end{align*}
\]

\[
\begin{align*}
\text{INTENTION} & \quad \text{EXECUTION} \\
\text{s} \quad \text{s} \quad \text{s} \quad \text{s} \quad \text{s} & \quad \text{s} \quad \text{s} \quad \text{s} \quad \text{s} \quad \text{s} \quad \text{s} \quad \text{s} \quad \text{s} \quad \text{s} \quad \text{s} \quad \text{s} \quad \text{s} \quad \text{s} \quad \text{s} \quad \text{s} \quad \text{s} \quad \text{s} \quad \text{s} \quad \text{s} \quad \text{s} \quad \text{s} & = 10
\end{align*}
\]
## Distance Matrix Computation

<table>
<thead>
<tr>
<th></th>
<th># 0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>#</td>
<td>E</td>
<td>X</td>
<td>E</td>
<td>C</td>
<td>U</td>
<td>T</td>
<td>I</td>
<td>O</td>
<td>N</td>
</tr>
</tbody>
</table>
### Distance Matrix

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>O</th>
<th>I</th>
<th>T</th>
<th>N</th>
<th>E</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Min of 4,6,6**
- **Min of 5,3,5**
- **Min of 2,2,2**
- **Min of 8,6,8**
## Distance Matrix

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>9</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>I</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>T</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>N</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>T</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>N</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>

# EXECUTION
Distance Matrix with shortest path

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>9</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>EX</td>
<td>E</td>
<td>C</td>
<td>U</td>
<td>T</td>
<td>I</td>
<td>O</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Another example

<table>
<thead>
<tr>
<th>R</th>
<th>I</th>
<th>G</th>
<th>H</th>
<th>T</th>
<th>R</th>
<th>I</th>
<th>T</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>I</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Edit Distance

9

<table>
<thead>
<tr>
<th>R</th>
<th>I</th>
<th>G</th>
<th>H</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>I</td>
<td>T</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>S</td>
<td>S</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

5

<table>
<thead>
<tr>
<th>R</th>
<th>I</th>
<th>G</th>
<th>H</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>I</td>
<td>T</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>D</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

3
Minimum Edit Distance Algorithm

• Create Matrix
• Initialize 1 – length in LH column and bottom row
• For each cell
  – Take the minimum of:
    • Deletion: +1 from left cell
    • Insertion: +1 from cell below
    • Substitution: Diagonal +0 if same +2 if different
  – Keep track of where you came from
Example

• Minimum of:
  – 1+1 (left right)
  – 1+1 (bottom up)
  – 0+0 (diagonal)

• Minimum of:
  – 0+1 (left right)
  – 2+1 (bottom up)
  – 1+2 (diagonal)
**Answer to Right-Rite**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>1</td>
<td>2, 0, 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>#</td>
<td>R</td>
<td>I</td>
<td>T</td>
<td>E</td>
</tr>
</tbody>
</table>

In each box X, Y, Z values are

- **X**: From left: Insert-add one from left box
- **Y**: Diagonal, Compare-0 if same, 2 if different
- **Z**: From below: Delete-add one from lower box

Minimum is highlighted in red with arrow to source

**NOTE**: All boxes will have arrows.

I didn’t show them all.

Only one path back to root.
### Answer to Right-Rite

<table>
<thead>
<tr>
<th>T</th>
<th>5</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>2</td>
<td>3, 3, 1</td>
<td>2, 0, 2</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>1</td>
<td>2, 0, 2</td>
<td>1, 3, 3</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>#</td>
<td>R</td>
<td>I</td>
<td>T</td>
<td>E</td>
</tr>
</tbody>
</table>

In each box X, Y, Z values are
- **X**: From left: Insert-add one from left box
- **Y**: Diagonal, Compare-0 if same, 2 if different
- **Z**: From below: Delete-add one from lower box

Minimum is highlighted in red with arrow to source

**NOTE**: All boxes will have arrows.
I didn’t show them all.
Only one path back to root.
## Answer to Right-Rite

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>6, 6, 4</th>
<th>5, 5, 5</th>
<th>6, 2, 4</th>
<th>3, 5, 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>4</td>
<td>5, 5, 3</td>
<td>4, 4, 2</td>
<td>3, 3, 3</td>
<td>4, 4, 4</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>4, 4, 2</td>
<td>3, 3, 1</td>
<td>2, 2, 2</td>
<td>3, 3, 3</td>
</tr>
<tr>
<td>I</td>
<td>2</td>
<td>3, 3, 1</td>
<td>2, 0, 2</td>
<td>1, 3, 3</td>
<td>2, 4, 4</td>
</tr>
<tr>
<td>R</td>
<td>1</td>
<td>2, 0, 2</td>
<td>1, 3, 3</td>
<td>2, 4, 4</td>
<td>3, 5, 5</td>
</tr>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>#</td>
<td></td>
<td>R</td>
<td>I</td>
<td>T</td>
<td>E</td>
</tr>
</tbody>
</table>

In each box X, Y, Z values are:

- **X**: From left: Insert-add one from left box
- **Y**: Diagonal, Compare-0 if same, 2 if different
- **Z**: From below: Delete-add one from lower box

Minimum is highlighted in red with arrow to source.

**NOTE:** All boxes will have arrows.

I didn’t show them all.

Only one path back to root.
## Answer to Right-Rite

<table>
<thead>
<tr>
<th>T</th>
<th>5</th>
<th>6, 6, 4</th>
<th>5, 5, 5</th>
<th>6, 2, 4</th>
<th>3, 5, 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>4</td>
<td>5, 5, 3</td>
<td>4, 4, 2</td>
<td>3, 3, 3</td>
<td>4, 4, 4</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>4, 4, 2</td>
<td>3, 3, 1</td>
<td>2, 2, 2</td>
<td>3, 3, 3</td>
</tr>
<tr>
<td>I</td>
<td>2</td>
<td>3, 3, 1</td>
<td>2, 0, 2</td>
<td>1, 3, 3</td>
<td>2, 4, 4</td>
</tr>
<tr>
<td>R</td>
<td>1</td>
<td>2, 0, 2</td>
<td>1, 3, 3</td>
<td>2, 4, 4</td>
<td>3, 5, 5</td>
</tr>
<tr>
<td>#</td>
<td>0</td>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

In each box X, Y, Z values are

- **X**: From left: Insert-add one from left box
- **Y**: Diagonal, Compare-0 if same, 2 if different
- **Z**: From below: Delete-add one from lower box

Minimum is highlighted in red with arrow to source

**NOTE**: All boxes will have arrows.

I didn’t show them all.

Only one path back to root.
• Minimum Edit Distance
• A “dynamic programming” algorithm
• We will see a probabilistic version of this called “Viterbi”