Accelerated Natural Language Processing
Lecture 3
Morphology and Finite State Machines;
Edit Distance

Sharon Goldwater
(based on slides by Philipp Koehn)

20 September 2018
Recap: Tasks

- Recognition
  - given: surface form
  - wanted: yes/no decision if it is in the language

- Generation
  - given: lemma and morphological properties
  - wanted: surface form

- Analysis
  - given: surface form
  - wanted: lemma and morphological properties
Recap: General approach

- Could list all words with their analyses, but
  - List gets too big
  - Language is infinite, cannot generalize beyond list

- Instead, use finite state machines
  - Finite and compact representation of infinite language
  - Several toolkits available
Recap: Finite State Automata

Can be viewed as either emitting or recognizing strings
Today’s lecture

• How are FSMs and FSTs used for morphological recognition, analysis and generation?

• How can we deal with spelling changes in morphological analysis?

• What is an alignment between two strings?

• What is minimum edit distance and how do we compute it? What’s wrong with a brute force solution, and how do we solve that problem?
One Word

Basic finite state automaton:

- start state
- transition that emits the word \textit{walk}
- end state

\[ S \xrightarrow{\text{walk}} E \]
One Word and One Inflection

Two transitions and intermediate state

- first transition emits \textit{walk}
- second transition emits \textit{+ed}

$\rightarrow$ \textit{walked}
One Word and Multiple Inflections

Multiple transitions between states

- three different paths

→ walks, walked, walking
Multiple Words and Multiple Inflections

Multiple stems
- implements regular verb morphology
  - laughs, laughed, laughing
  - walks, walked, walking
  - reports, reported, reporting
Multiple Words and Multiple Inflections

Multiple stems
- implements regular verb morphology
  → what about bake, emit, fuss?
more on this later...
Derivational Morphology

word

START

double lines = end state
Derivational Morphology

START → word → y → START
Derivational Morphology

again: \textit{wordify} not \textit{wordyfy}!

again, will come back to that later...
Derivational Morphology

why a loop? could it be placed differently?
Derivational Morphology

START

word

ISM

ist

cy

fy

cate

ion

er
Marking Part of Speech

START

word

NN

y

fy

ion

cate

NN

NN

NN

NN

NN
Marking Part of Speech

Now: where to add -less? -ness? Others?

Sharon Goldwater  ANLP Lecture 3  16
Concatenation

- Constructing an FSA gets very complicated

- Build components as separate FSAs
  - L: FSA for lexicon
  - D: FSA for derivational morphology
  - I: FSA for inflectional morphology

- Concatenate L + D + I (there are standard algorithms)
  - In fact, each component may consist of multiple components (e.g., D has different sets of affixes with ordering constraints)
What is Required?

- Lexicon of lemmas
  - very large, needs to be collected by hand
- Inflection and derivation rules
  - not large, but requires understanding of the language

Recent work: automatically learn lemmas and suffixes from a corpus

- OK solution for languages with few resources
- Hand-engineered systems much better when available
Generation and analysis

• FSAs used as morphological recognizers

• What if we want to generate or analyze?

  \[ \text{walk} + V + \text{past} \leftrightarrow \text{walked} \]
  \[ \text{report} + V + \text{prog} \leftrightarrow \text{reporting} \]

• Use a finite-state transducer (FST)
  – Replace output symbols with input-output pairs \( x : y \)
FSA for verbs

laugh
walk
report
s
ed
ing
Schematically

\[ \text{verb-reg} \rightarrow \text{ed} \rightarrow \text{s} \]

\[ \text{ing} \rightarrow \text{ed} \rightarrow \text{s} \]
FST for verbs

where $x$ means $x:x$ and $x:\epsilon$ means $x:\epsilon$. 
Accounting for spelling changes

• We now have:

\[
\text{walk} + V + \text{past} \leftrightarrow \text{walked} \\
\text{BUT} \\
\text{bake} + V + \text{past} \leftrightarrow \text{bakeed}
\]

• How to fix this?
Accounting for spelling changes

- We now have:

  \[
  \text{walk} + V + \text{past} \leftrightarrow \text{walked} \\
  \text{BUT} \\
  \text{bake} + V + \text{past} \leftrightarrow \text{bakeed}
  \]

- How to fix this? Use two FSTs in a row!

  \[
  \text{walk} + V + \text{past} \leftrightarrow \text{walk}^\text{ed}\# \leftrightarrow \text{walked} \\
  \text{bake} + V + \text{past} \leftrightarrow \text{bake}^\text{ed}\# \leftrightarrow \text{baked}
  \]
1. **Analysis to intermediate form**

where \( x \) means \( x:x \) and \( x: \) means \( x:\epsilon \).

- **Examples:**
  - walk\(^V\)+past \(\leftrightarrow\) walk\(^\text{ed}\)\#
  - bake\(^V\)+past \(\leftrightarrow\) bake\(^\text{ed}\)\#
  - bake\(^V\)+prog \(\leftrightarrow\) bake\(^\text{ing}\)#
2. Intermediate form to surface form

Simplified version, only handles some aspects of past tense:

other^:ed#::ed

where other means any character except ‘e’.

- Examples: walk^ed# ↔ walked, bake^ed# ↔ baked

- A nondeterministic FST: multiple transitions may be possible on the same input (where?). If any path goes to end state, string is accepted.
Plural transducer (J&M, Fig. 3.17)

- Complete FST for English plural (‘other’ = none of \{z,s,x,^,#,\epsilon\})
- What happens in each case?  cat\^s#  fox\^s#  axle\^s#
Remaining problem: ambiguity

• FSTs often produce multiple analyses for a single form:

\[
w \text{walks} \rightarrow \text{walk} + V + 1\text{sg} \ \text{OR} \ \text{walk} + N + \text{pl}
\]

German ‘the’: 6 surface forms, but 24 possible analyses

• Resolve using context (surrounding words), usually in a probabilistic system (stay tuned...)

Sharon Goldwater  ANLP Lecture 3  28
More info and tools


- OpenFST (Google and NYU)
  http://www.openfst.org/

- Carmel Toolkit
  http://www.isi.edu/licensed-sw/carmel/

- FSA Toolkit
  http://www-i6.informatik.rwth-aachen.de/~kanthak/fsa.html
Related task: string similarity

Given two strings, how “similar” are they?

- Could indicate morphological relationships:
  
  walk - walks, sleep - slept
  
- Or possible spelling errors (and corrections):
  
  definition - defintion, separate - seperate
  
- Also used in other fields, e.g., bioinformatics:
  
  ACCGTA - ACCGATA
One measure: minimum edit distance

• How many changes to go from string $s_1 \rightarrow s_2$?

  \begin{array}{cccccc}
  S & T & A & L & L \\
  T & A & L & L & \text{deletion} \\
  T & A & B & L & \text{substitution} \\
  T & A & B & L & E & \text{insertion} \\
  \end{array}

• To solve the problem, we need to find the best alignment between the words.
  
  – Could be several equally good alignments.
Example alignments

Let ins/del cost (distance) = 1, sub cost = 2 (0 if no change) (can use other costs, incl diff costs for diff chars)

- Two optimal alignments (cost = 4):

\[
\begin{array}{ccccc}
S & T & A & L & L - \\
d & & s & i & \\
- & T & A & B & L & E
\end{array}
\quad
\begin{array}{ccccc}
S & T & A & - & L & L \\
d & & i & s & \\
- & T & A & B & L & E
\end{array}
\]
Example alignments

Let ins/del cost (distance) = 1, sub cost = 2 (0 if no change) (can use other costs, incl diff costs for diff chars)

• Two optimal alignments (cost = 4):

\[
\begin{align*}
S & \enspace T & A & \enspace L & \enspace L & - & S & \enspace T & A & \enspace - & L & L \\
d & | & | & s & | & i & d & | & | & i & | & s \\
- & T & A & B & L & E & - & T & A & B & L & E
\end{align*}
\]

• LOTS of non-optimal alignments, such as:

\[
\begin{align*}
S & \enspace T & A & \enspace - & L & - & L & S & \enspace T & A & L & - & L & - \\
s & d & | & i & | & i & d & d & d & s & s & i & | & i \\
\end{align*}
\]
Brute force solution: too slow

How many possible alignments to consider?

• First character could align to any of:

- - - - - - T A B L E -

• Next character can align anywhere to its right

• And so on... the number of alignments grows exponentially with the length of the sequences.
**Brute force solution: too slow**

How many possible alignments to consider?

- First character could align to any of:
  
  - - - - - T A B L E -

- Next character can align anywhere to its right

- And so on... the number of alignments grows exponentially with the length of the sequences.

To solve, we use a *dynamic programming* algorithm

- Store solutions to smaller computations and combine them

- Widespread in NLP, e.g. tagging (HMMs), parsing (CKY)
Intuition

• Minimum distance $D(\text{stall, table})$ must be the minimum of:
  
  – $D(\text{stall, tabl}) + 1$ (ins)
  – $D(\text{stal, table}) + 1$ (del)
  – $D(\text{stal, tabl}) + 2$ (sub)

• Similarly for the smaller subproblems

• So proceed as follows:
  
  – solve smallest subproblems first
  – store solutions in a table (chart)
  – use these to solve and store larger subproblems until we get the full solution
### Chart: starting point

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>A</th>
<th>B</th>
<th>L</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Chart\([i, j]\) stores \(D(\text{stall}[0..i], \text{table}[0..j])\)
  - Ex: **Chart[3,0]** = \(D(\text{stall}[0..3], \text{table}[0..0]) = D(\text{sta}, \epsilon)\)
Chart: next step

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>A</th>
<th>B</th>
<th>L</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Chart[1, 1]** is $D(S, T)$: the minimum of
  
  $D(-, T) + 1$ (Chart[0, 1] + 1 = 2)
  
  $D(S, -) + 1$ (Chart[1, 0] + 1 = 2)
  
  $D(-, -) + 2$ (Chart[0, 0] + 2 = 2)
## Chart: one more step

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>A</th>
<th>B</th>
<th>L</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

- **Chart[2, 1]** is $D(ST, T)$: the minimum of
  - $D(S, T) + 1$ \((\text{Chart[1, 1]} + 1 = 3)\)
  - $D(ST, -) + 1$ \((\text{Chart[2, 0]} + 1 = 3)\)
  - $D(S, -) + 0$ \((\text{Chart[1, 0]} + 0 = 1)\)
<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>A</th>
<th>B</th>
<th>L</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Continue by filling in each full column in order (or go by rows)
## Chart: completed

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>A</th>
<th>B</th>
<th>L</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>L</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>L</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>
To find alignments

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>A</th>
<th>B</th>
<th>L</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>←1</td>
<td>←2</td>
<td>←3</td>
<td>←4</td>
<td>←5</td>
</tr>
<tr>
<td>S</td>
<td>↑1</td>
<td>←↖↑2</td>
<td>←↖↑3</td>
<td>←↖↑4</td>
<td>←↖↑5</td>
</tr>
<tr>
<td>T</td>
<td>↑2</td>
<td>←1</td>
<td>←2</td>
<td>←3</td>
<td>←4</td>
</tr>
<tr>
<td>A</td>
<td>↑3</td>
<td>↑2</td>
<td>←1</td>
<td>←2</td>
<td>←3</td>
</tr>
<tr>
<td>L</td>
<td>↑4</td>
<td>↑3</td>
<td>↑2</td>
<td>←↖↑3</td>
<td>←2</td>
</tr>
<tr>
<td>L</td>
<td>↑5</td>
<td>↑4</td>
<td>↑3</td>
<td>←↖↑4</td>
<td>←↑3</td>
</tr>
</tbody>
</table>

- also store *which subproblem* the best score came from
- backtrack to get the best alignment

⇒ More complete worked example on lecture page, with exercises.
Questions for review

• How are FSMs and FSTs used for morphological recognition, analysis and generation?

• How can we deal with spelling changes in morphological analysis?

• What is an alignment between two strings?

• What is minimum edit distance and how do we compute it? What’s wrong with a brute force solution, and how do we solve that problem?
Announcements

• Next lecture: Probability models and estimation.
  – Assumes you know or are getting to grips with the material in the maths tutorials on the Readings section.
  – Start working on the tutorial exercise sheets (see web page): bring questions to your tutorial group.

• Tutorials start as early as Tuesday. Register your availability by today so we can assign groups! Use link on Learn, please enter more than one choice if you can.

• While there, register for Piazza (use link on sidebar).

• TA hours start next week, see Learn for times (under Timetable).