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 walk
- end state

One Word and One Inflection

Two transitions and intermediate state
- first transition emits walk
- second transition emits +ed
→ 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
again: wordify not wordyfy!
again, will come back to that later...

why a loop? could it be placed differently?
Marking Part of Speech

now: where to add -less? -ness? others?

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

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)

generation and analysis

- FSAs used as morphological recognizers
- what if we want to generate or analyze?
walk+V+past ↔ walked
report+V+prog ↔ reporting
- use a finite-state transducer (FST)
  - replace output symbols with input-output pairs $x : y$
FSA for verbs

Schematically

FST for verbs

Accounting for spelling changes

- We now have:
  walk+V+past ↔ walked
  BUT
  bake+V+past ↔ bakeed

- How to fix this?

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

- We now have:
  \[
  \text{walk}+V+\text{past} \leftrightarrow \text{walked} \\
  \text{bake}+V+\text{past} \leftrightarrow \text{bakeed}
  \]

- How to fix this? Use two FSTs in a row!
  \[
  \text{walk}+V+\text{past} \leftrightarrow \text{walk}^\ddag \text{ed}\# \leftrightarrow \text{walked} \\
  \text{bake}+V+\text{past} \leftrightarrow \text{bake}^\ddag \text{ed}\# \leftrightarrow \text{baked}
  \]

1. Analysis to intermediate form

\[
\text{verb-reg} + V:^ \rightarrow + 3\text{sg}:s^# \rightarrow + \text{past}:\text{ed}\# \rightarrow + \text{prog}:\text{ing}\#
\]

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

- Examples:
  \[
  \text{walk}+V+\text{past} \leftrightarrow \text{walk}^\ddag \text{ed}\# \\
  \text{bake}+V+\text{past} \leftrightarrow \text{bake}^\ddag \text{ed}\# \\
  \text{bake}+V+\text{prog} \leftrightarrow \text{bake}^\ddag \text{ing}\#
  \]

2. Intermediate form to surface form

Simplified version, only handles some aspects of past tense:

\[
\text{e,other} \rightarrow \text{other} \rightarrow ^\ddag \rightarrow \text{ed}^\# \rightarrow \text{ed}
\]

where other means any character except 'e'.

- Examples: \(\text{walk}^\ddag \text{ed}\# \leftrightarrow \text{walked}\), \(\text{bake}^\ddag \text{ed}\# \leftrightarrow \text{baked}\)

- 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? \(\text{cat}^\ddag \text{s}\#\), \(\text{fox}^\ddag \text{s}\#\), \(\text{axle}^\ddag \text{s}\#\)
Remaining problem: ambiguity

- FSTs often produce multiple analyses for a single form:
  walks → walk+V+1sg OR walk+N+pl
  German ‘the’: 6 surface forms, but 24 possible analyses
- Resolve using context (surrounding words), usually in a probabilistic system (stay tuned...)

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 $s_1 \rightarrow s_2$?

  S T A L L
  T A L L deletion
  T A B L substitution
  T A B L E insertion

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

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

- Lots of non-optimal alignments, such as:
  \[
  \begin{align*}
  &S T A - L - L \\
  &\quad s \quad d \quad | \quad i \quad | \quad i \quad d \\
  &\quad - T A B L E \\
  &S T A L - L - \\
  &\quad d \quad d \quad s \quad s \quad i \quad | \quad i \\
  &\quad - - T A B L E
  \end{align*}
  \]

Brute force solution: too slow

How many possible alignments to consider?

- First character could align to any of:
  \[
  \begin{align*}
  &- - - - - T A B L E - \\
  \end{align*}
  \]

- Next character can align anywhere to its right
- And so on... the number of alignments grows exponentially with the length of the sequences.

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

- Lots of non-optimal alignments, such as:
  \[
  \begin{align*}
  &S T A - L - L \\
  &\quad s \quad d \quad | \quad i \quad | \quad i \quad d \\
  &\quad - T A B L E \\
  &S T A L - L - \\
  &\quad d \quad d \quad s \quad s \quad i \quad | \quad i \\
  &\quad - - T A B L E
  \end{align*}
  \]

Brute force solution: too slow

How many possible alignments to consider?

- First character could align to any of:
  \[
  \begin{align*}
  &- - - - - T A B L E - \\
  \end{align*}
  \]

- 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:
  - $D(\text{stall, table}) + 1$ (ins)
  - $D(\text{stall, table}) + 1$ (del)
  - $D(\text{stall, table}) + 2$ (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 $D(\text{S, T})$: the minimum of
  - $D(-, T) + 1$ (Chart$[0, 1] + 1 = 3$)
  - $D(\text{S, }-) + 1$ (Chart$[1, 0] + 1 = 2$)
  - $D(-, -) + 0$ (Chart$[0, 0] + 0 = 1$)

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>
<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>?</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$[2, 1]$ is $D(\text{ST, T})$: the minimum of
  - $D(\text{S, T}) + 1$ (Chart$[1, 1] + 1 = 3$)
  - $D(\text{ST, }-) + 1$ (Chart$[2, 0] + 1 = 3$)
  - $D(\text{S, -}) + 0$ (Chart$[1, 0] + 0 = 1$)
• Continue by filling in each full column in order (or go by rows)

To find alignments

• 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).