Basic Text Processing

Regular Expressions
Regular expressions

- A formal language for specifying text strings
- How can we search for any of these?
  - woodchuck
  - woodchucks
  - Woodchuck
  - Woodchucks
Regular Expressions: Disjunctions

- Letters inside square brackets []

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>[wW]oodchuck</td>
<td>Woodchuck, woodchuck</td>
</tr>
<tr>
<td>[1234567890]</td>
<td>Any digit</td>
</tr>
</tbody>
</table>

- Ranges [A–Z]

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A–Z]</td>
<td>An upper case letter</td>
</tr>
<tr>
<td>[a–z]</td>
<td>A lower case letter</td>
</tr>
<tr>
<td>[0–9]</td>
<td>A single digit</td>
</tr>
</tbody>
</table>

Drenched Blossoms

Chapter 1: Down the Rabbit Hole
# Regular Expressions: Negation in Disjunction

- **Negations** \[^Ss\]
  - Caret means negation only when first in []

## Pattern Matches

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
</table>
| \[^A-Z\] | Not an upper case letter | *Oyfn pripeitchik*
| \[^Ss\]  | Neither ‘S’ nor ‘s’ | *I have no exquisite reason”*
| \[^e^\]  | Neither e nor ^ | *Look here*
| \(a^b\) | The pattern a caret b | *Look up \(a^b\) now*
Regular Expressions: More Disjunction

- Woodchucks is another name for groundhog!
- The pipe | for disjunction

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>groundhog</td>
<td>woodchuck</td>
</tr>
<tr>
<td>yours</td>
<td>mine</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>[gG]roundhog</td>
<td>[Ww]oodchuck</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>colou?r</td>
<td>Optional previous char</td>
</tr>
<tr>
<td>oo*h!</td>
<td>0 or more of previous char</td>
</tr>
<tr>
<td>o+h!</td>
<td>1 or more of previous char</td>
</tr>
<tr>
<td>baa+</td>
<td></td>
</tr>
<tr>
<td>beg.n</td>
<td></td>
</tr>
</tbody>
</table>

*Stephen C Kleene*

Kleene *,  Kleene +
## Regular Expressions: Anchors

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>^[A-Z]</td>
<td>Palo Alto</td>
</tr>
<tr>
<td>^[^A-Za-z]</td>
<td>Hello”</td>
</tr>
<tr>
<td>.$</td>
<td>The end</td>
</tr>
<tr>
<td>.$</td>
<td>The end? The end!</td>
</tr>
</tbody>
</table>
Example

- Find me all instances of the word “the” in a text.
  
  \texttt{the}

  \textit{Misses capitalized examples}

  \texttt{[tT]he}

  \textit{Incorrectly returns other or theology}

  \texttt{[^a-zA-Z][tT]he[^a-zA-Z]}
Errors

The process we just went through was based on fixing two kinds of errors:

- Matching strings that we should not have matched (there, then, other)
  - False positives (Type I)
- Not matching things that we should have matched (The)
  - False negatives (Type II)
In NLP we are always dealing with these kinds of errors.

Reducing the error rate for an application often involves two antagonistic efforts:

- Increasing accuracy or precision (minimizing false positives)
- Increasing coverage or recall (minimizing false negatives).
Summary

- Regular expressions play a surprisingly large role
  - Sophisticated sequences of regular expressions are often the first model for any text processing
- For many hard tasks, we use machine learning classifiers
  - But regular expressions are used as features in the classifiers
  - Can be very useful in capturing generalizations
Basic Text Processing

Word tokenization
Text Normalization

- Every NLP task needs to do text normalization:
  1. Segmenting/tokenizing words in running text
  2. Normalizing word formats
  3. Segmenting sentences in running text
I do uh main- mainly business data processing
   Fragments, filled pauses
Seuss’s **cat** in the hat is different from other **cats**!
   **Lemma**: same stem, part of speech, rough word sense
      - cat and cats = same lemma
   **Wordform**: the full inflected surface form
      - cat and cats = different wordforms

How many words?
How many words?

they lay back on the San Francisco grass and looked at the stars and their

- **Type**: an element of the vocabulary.
- **Token**: an instance of that type in running text.
- How many?
  - 15 tokens (or 14)
  - 13 types (or 12)
How many words?

\( N = \text{number of tokens} \)

\( V = \text{vocabulary} = \text{set of types} \)

\( |V| \) is the size of the vocabulary

Church and Gale (1990): \( |V| > O(N^{\frac{1}{2}}) \)

|                      | Tokens = N         | Types = |V| |
|----------------------|--------------------|---------|----|
| Switchboard phone   | 2.4 million        | 20 thousand |
| conversations       |                    |         |
| Shakespeare          | 884,000            | 31 thousand |
| Google N-grams       | 1 trillion         | 13 million |
Simple Tokenization in UNIX

• (Inspired by Ken Church’s UNIX for Poets.)
• Given a text file, output the word tokens and their frequencies

```
tr -sc 'A-Za-z' '
' < shakes.txt | sort | uniq -c
```

1945 A
72 AARON
19 ABBESS
5 ABBOT
...

25 Aaron
6 Abate
1 Abates
5 Abbess
6 Abbey
3 Abbot
...

Change all non-alpha to newlines
Sort in alphabetical order
Merge and count each type
Issues in Tokenization

- Finland’s capital → Finland Finlands Finland’s ?
- what’re, I’m, isn’t → What are, I am, is not
- Hewlett-Packard → Hewlett Packard ?
- state-of-the-art → state of the art ?
- Lowercase → lower-case lowercase lower case ?
- San Francisco → one token or two?
- m.p.h., PhD. → ??
Tokenization: language issues

- French
  - *L'ensemble* → one token or two?
    - *L ? L’ ? Le ?*
    - Want *l’ensemble* to match with *un ensemble*

- German noun compounds are not segmented
  - *Lebensversicherungsgesellschaftsangestellter*
  - ‘life insurance company employee’
  - German information retrieval needs *compound splitter*
Tokenization: language issues

- Chinese and Japanese no spaces between words:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
  - Sharapova now lives in US southeastern Florida

- Further complicated in Japanese, with multiple alphabets intermingled
  - Dates/amounts in multiple formats

フォーチュン500社は情報不足のため時間あた$500K(約6,000万円)

End-user can express query entirely in hiragana!
Word Tokenization in Chinese

- Also called **Word Segmentation**
- Chinese words are composed of characters
  - Characters are generally 1 syllable and 1 morpheme.
  - Average word is 2.4 characters long.
- Standard baseline segmentation algorithm:
  - Maximum Matching (also called Greedy)
Maximum Matching
Word Segmentation Algorithm

- 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
Max-match segmentation illustration

- The cat in the hat
- The table down there
- Doesn’t generally work in English!
- But works astonishingly well in Chinese
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
- Modern probabilistic segmentation algorithms even better
Basic Text Processing

Word Normalization and Stemming
Normalization

- Need to “normalize” terms
  - Information Retrieval: indexed text & query terms must have same form.
  - We want to match *U.S.A.* and *USA*
- We implicitly define equivalence classes of terms
  - e.g., deleting periods in a term
- Alternative: asymmetric expansion:
  - Enter: *window*  Search: *window, windows*
  - Enter: *windows*  Search: *Windows, windows, window*
  - Enter: *Windows*  Search: *Windows*
Case folding

- Applications like IR: reduce all letters to lower case
  - Since users tend to use lower case
  - Possible exception: upper case in mid-sentence?
    - e.g., General Motors
    - Fed vs. fed
    - SAIL vs. sail

- For sentiment analysis, MT, Information extraction
  - Case is helpful (US versus us is important)
Lemmatization

- Reduce inflections or variant forms to base form
  - *am, are, is* → *be*
  - *car, cars, car's, cars'* → *car*
  - *the boy's cars are different colors* → *the boy car be different color*
- Lemmatization: have to find correct dictionary headword form
- Machine translation
  - Spanish *quiero* (‘I want’), *quieres* (‘you want’) same lemma as *querer* ‘want’
Morphology

- **Morphemes**: The small meaningful units that make up words
- **Stems**: The core meaning-bearing units
- **Affixes**: Bits and pieces that adhere to stems
  - Often with grammatical functions
Stemming

- Reduce terms to their stems in information retrieval
- *Stemming* is crude chopping of affixes
  - language dependent
  - e.g., *automate(s), automatic, automation* all reduced to *automat*.

*For example compressed and compression are both accepted as equivalent to compress.*

*For example, compress and compress are both accepted as equivalent to compress.*
Porter’s algorithm
The most common English stemmer

Step 1a

sses → ss  caresses → caress
ies → i  ponies → poni
ss → ss  caress → caress
s → Ø  cats → cat

Step 1b

(*v*)ing → Ø  walking → walk
sing → sing

(*v*)ed → Ø  plastered → plaster

Step 2 (for long stems)

ational → ate  relational → relate
izer → ize  digitizer → digitize
ator → ate  operator → operate
...

Step 3 (for longer stems)

al → Ø  revival → reviv
able → Ø  adjustable → adjust
ate → Ø  activate → activ
...

...
Viewing morphology in a corpus
Why only strip -ing if there is a vowel?

(*v*)ing → Ø   walking → walk
   sing → sing

```
tr -sc 'A-Za-z' '
' < shakes.txt | grep 'ing$' | sort | uniq -c | sort -nr

1312 King
548 being
541 nothing
388 king
375 bring
358 thing
307 ring
152 something
145 coming
130 morning
122 having
120 living
117 loving
116 Being
102 going
```

```
tr -sc 'A-Za-z' '
' < shakes.txt | grep '[aeiou].*ing$' | sort | uniq -c | sort -nr
```
Dealing with complex morphology is sometimes necessary

- Some languages require complex morpheme segmentation
  - Turkish
  - Uygarlastiradamiklerimizdanmissinizcasina
  - `(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’
Basic Text Processing

Sentence Segmentation and Decision Trees
Sentence Segmentation

- !, ? are relatively unambiguous
- Period “.” is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02% or 4.3
- Build a binary classifier
  - Looks at a “.”
  - Decides EndOfSentence/NotEndOfSentence
  - Classifiers: hand-written rules, regular expressions, or machine-learning
Determining if a word is end-of-sentence: a Decision Tree

- **Lots of blank lines after me?**
  - **YES**
    - **Final punctuation is ?, !, or ?:**
      - **YES**
        - **Final punctuation is period**
        - **YES**
          - **I am “etc” or other abbreviation**
            - **YES**
              - **Not E-O-S**
            - **NO**
              - **Not E-O-S**
          - **NO**
            - **Not E-O-S**
        - **NO**
          - **E-O-S**
    - **NO**
      - **E-O-S**

- **NO**
  - **Final punctuation is period**
    - **YES**
      - **I am “etc” or other abbreviation**
        - **YES**
          - **Not E-O-S**
        - **NO**
          - **Not E-O-S**
    - **NO**
      - **E-O-S**
More sophisticated decision tree features

- Case of word with “.”: Upper, Lower, Cap, Number
- Case of word after “.”: Upper, Lower, Cap, Number

- Numeric features
  - Length of word with “.”
  - Probability(word with “.” occurs at end-of-s)
  - Probability(word after “.” occurs at beginning-of-s)
Implementing Decision Trees

- A decision tree is just an if-then-else statement
- The interesting research is choosing the features
- Setting up the structure is often too hard to do by hand
  - Hand-building only possible for very simple features, domains
    - For numeric features, it’s too hard to pick each threshold
  - Instead, structure usually learned by machine learning from a training corpus
Words and the Company They Keep

Firth, J. R. 1957:11
Motivation

- Environment:
  - Mostly “not a full analysis (sentence/text parsing)”

- Tasks where “words & company” are important:
  - Word sense disambiguation (MT, IR, TD, IE)
  - Lexical entries: subdivision & definitions (lexicography)
  - Language modeling (generalization, [kind of] smoothing)
  - Word/phrase/term translation (MT, Multilingual IR)
  - NL generation (“natural” phrases) (Generation, MT)
  - Parsing (lexically-based selectional preferences)
Collocations

- Collocation
  - Firth: “word is characterized by the company it keeps”; collocations of a given word are statements of the habitual or customary places of that word.
  - non-compositionality of meaning
    - cannot be derived directly from its parts (heavy rain)
  - non-substitutability in context
    - for parts (red light)
  - non-modifiability (＆non-transformability)
    - kick the yellow bucket; take exceptions to
Association and Co-occurrence; Terms

- Does not fall under “collocation”, but:
- Interesting just because it does often [rarely] appear together or in the same (or similar) context:
  - (doctors, nurses)
  - (hardware, software)
  - (gas, fuel)
  - (hammer, nail)
  - (communism, free speech)
Collocations of Special Interest

- Idioms: really fixed phrases
  - kick the bucket, birds-of-a-feather, run for office

- Proper names: difficult to recognize even with lists
  - Tuesday (person’s name), May, Winston Churchill, IBM, Inc.

- Numerical expressions
  - containing “ordinary” words
    - Monday Oct 04 1999, two thousand seven hundred fifty

- Phrasal verbs
  - Separable parts:
    - look up, take off
Further Notions

- **Synonymy**: different form/word, same meaning:
  - notebook / laptop

- **Antonymy**: opposite meaning:
  - new/old, black/white, start/stop

- **Homonymy**: same form/word, different meaning:
  - “true” (random, unrelated): can (aux. verb / can of Coke)
  - related: polysemy; notebook, shift, grade, …

- **Other**:
  - Hyperonymy/Hyponymy: general vs. special: vehicle/car
  - Meronymy/Holonomy: whole vs. part: body/leg
How to Find Collocations?

- Frequency
  - plain
  - filtered
- Hypothesis testing
  - $t$ test
  - $\chi^2$ test
- Pointwise ("poor man’s") Mutual Information
- (Average) Mutual Information
Frequency

- Simple
  - Count n-grams; high frequency n-grams are candidates:
    - mostly function words
    - frequent names

- Filtered
  - Stop list: words/forms which (we think) cannot be a part of a collocation
    - a, the, and, or, but, not, …
  - Part of Speech (possible collocation patterns)
    - A+N, N+N, N+of+N, …
Hypothesis Testing

- **Hypothesis**
  - something we test (against)

- **Most often:**
  - compare possibly interesting thing vs. “random” chance
  - “Null hypothesis”:
    - something occurs by chance (that’s what we suppose).
    - Assuming this, prove that the probability of the “real world” is then too low (typically < 0.05, also 0.005, 0.001)… therefore reject the null hypothesis (thus confirming “interesting” things are happening!)
  - Otherwise, it’s possible there is nothing interesting.
**t test (Student’s t test)**

- **Significance of difference**
  - compute “magic” number against normal distribution (mean $\mu$)
  - using real-world data: ($x'$ real data mean, $s^2$ variance, $N$ size):
    - $t = \frac{(x' - \mu)}{\sqrt{s^2 / N}}$
  - find in tables (see MS, p. 609):
    - $d.f.$ = degrees of freedom (parameters which are not determined by other parameters)
    - percentile level $p = 0.05$ (or better) (90% confidence; double tail)
  - the bigger $t$:
    - the better chances that there is the interesting feature we hope for (i.e. we can reject the null hypothesis)
    - $t$: at least the value from the table(s)
**t test on words**

- **null hypothesis:** independence \( p(w_1, w_2) = p(w_1)p(w_2) \)
  - mean \( \mu \): \( p(w_1)p(w_2) \)

- **data estimates:**
  - \( x' \) = MLE of joint probability from data
  - \( s^2 \) is \( p(1-p) \), i.e. almost \( p \) for small \( p \); \( N \) is the data size

**Example: (d.f. ~ sample size)**

- ‘general term’ (in corpus): \( c(\text{general}) = 108, c(\text{term}) = 40 \)
- \( c(\text{general}, \text{term}) = 2 \); expected \( p(\text{general})p(\text{term}) = 8.8E-8 \)
- \( t = \frac{(9.0E-6 - 8.8E-8)}{(9.0E-6 / 221097)^{1/2}} = 1.40 \) (not > 2.576) thus ‘general term’ is **not** a collocation with confidence 0.005 (99%) (not even with 0.05(90%))
- ‘true species’: \( (84/1779/9) \): \( t = 2.774 > 2.576 !! \)
Pearson’s Chi-square test

- $\chi^2$ test (general formula): $\sum_{i,j} (O_{ij} - E_{ij})^2 / E_{ij}$
  - where $O_{ij} / E_{ij}$ is the observed/expected count of events $i, j$
  - for two-outcomes-only events:

<table>
<thead>
<tr>
<th>$w_{\text{right}} \backslash w_{\text{left}}$</th>
<th>= true</th>
<th>$\neq$ true</th>
</tr>
</thead>
<tbody>
<tr>
<td>$= \text{species}$</td>
<td>9</td>
<td>1,770</td>
</tr>
<tr>
<td>$\neq \text{species}$</td>
<td>75</td>
<td>219,243</td>
</tr>
</tbody>
</table>

$\chi^2 = 221097(219243\times9 - 75\times1770)^2 / 1779\times84\times221013\times219318$ (formula in MS book) = 103.39 > 7.88 (at .005 thus we can reject the independence assumption) (OK with even .001)

$$X^2 = N(O_{11}O_{22} - O_{12}O_{21})^2 / (O_{11}+O_{12})(O_{11}+O_{21})(O_{12}+O_{22})(O_{21}+O_{22})$$
Pointwise Mutual Information

- This is **NOT** the MI as defined in Information Theory
  - (IT: average of the following; not of *values*)
- ...but might be useful:
  \[ I'(a,b) = \log_2 \left( \frac{p(a,b)}{p(a)p(b)} \right) = \log_2 \left( \frac{p(a|b)}{p(a)} \right) \]
- Example (same):
  \[
  I'(true, species) = \log_2 \left( \frac{4.1\times10^{-5}}{3.8\times10^{-4} \times 8.0\times10^{-3}} \right) = 3.74
  
  I'(general, term) = \log_2 \left( \frac{9.0\times10^{-6}}{1.8\times10^{-4} \times 4.9\times10^{-4}} \right) = 6.68
  \]
- measured in bits but it is difficult to give it an interpretation
- used for ranking (~ the null hypothesis tests)
- No good for sparse data (good for independence, but not for dependence)
- Dunning's Likelihood ratio is better especially for sparse data (see 172p in MS book) (log likelihood ratio between two hypotheses)
  \[-2\log \lambda = \chi^2 \text{ distribution; } \lambda = \text{likelihood of two hypotheses ratio } h_1=\text{word1} \]
  \[ , \text{ word2 occurrence independent; } h_2=\text{word1, word2 dependent} \text{ (likelihood of } h_1/h_2 \text{ using binomial distribution) } \]
Minimum Edit Distance
How similar are two strings?

• Spell correction
  • The user typed “graffe”
    Which is closest?
    • graf
    • graft
    • grail
    • giraffe

• Computational Biology
  • Align two sequences of nucleotides

    AGGCTATCACCTGACCTCCAGGCCGATGCCC
    TAGCTATCACGACC CGGTCGATTTGCCCCGAC
  • Resulting alignment:

    ─AGGCTATCACCTGACCTCCAGGCCGAA--TGCCC---
    TAGCTATCACGACC CGGTCGATTTGCCCCGAC

• Also for Machine Translation, Information Extraction, Speech Recognition
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

**INTENTION**

*EXECUTION*

- If each operation has cost of 1
  - Distance between these is 5
- If substitutions cost 2 (Levenshtein)
  - Distance between them is 8
Alignment in Computational Biology

• Given a sequence of bases

\[
\begin{align*}
AGGCTATCACCTGACCTCCCAGGCCGATGCCC \\
TAGCTATCACGACCACGGCGGTGATTTGTCCCAGAC
\end{align*}
\]

• An alignment:

\[
\begin{align*}
-AGGCTATCACCTGACCTCCCAGGCCGATGCCC- \\
TAGCTATCACGACCACGGCGGTGATTTGTCCCAGAC
\end{align*}
\]

• Given two sequences, align each letter to a letter or gap
Other uses of Edit Distance in NLP

• Evaluating Machine Translation and speech recognition

R Spokesman confirms senior government adviser was shot
H Spokesman said the senior adviser was shot dead

        S       I            D          I

• Named Entity Extraction and Entity Coreference
  • IBM Inc. announced today
  • IBM profits
  • Stanford President John Hennessy announced yesterday
  • for Stanford University President John Hennessy
How to find the Min Edit Distance?

• Searching for a path (sequence of edits) from the start string to the final string:
  • **Initial state**: the word we’re transforming
  • **Operators**: insert, delete, substitute
  • **Goal state**: the word we’re trying to get to
  • **Path cost**: what we want to minimize: the number of edits
Defining Min Edit Distance

• For two strings
  • X of length \( n \)
  • Y of length \( m \)
• We define \( D(i,j) \)
  • the edit distance between \( X[1..i] \) and \( Y[1..j] \)
    • i.e., the first \( i \) characters of \( X \) and the first \( j \) characters of \( Y \)
  • The edit distance between \( X \) and \( Y \) is thus \( D(n,m) \)
Dynamic Programming for Minimum Edit Distance

- **Dynamic programming**: A tabular computation of $D(n,m)$
- Solving problems by combining solutions to subproblems.
- Bottom-up
  - We compute $D(i,j)$ for small $i,j$
  - And compute larger $D(i,j)$ based on previously computed smaller values
  - i.e., compute $D(i,j)$ for all $i (0 < i < n)$ and $j (0 < j < m)$
Defining Min Edit Distance (Levenshtein)

- **Initialization**
  
  \[
  D(i,0) = i \\
  D(0,j) = j
  \]

- **Recurrence Relation:**
  
  For each \( i = 1..M \)
  
  For each \( j = 1..N \)
  
  \[
  D(i,j) = \min \begin{cases} 
  D(i-1,j) + 1 \\
  D(i,j-1) + 1 \\
  D(i-1,j-1) + 2; \text{ if } X(i) \neq Y(j) \\
  0; \text{ if } X(i) = Y(j)
  \end{cases}
  \]

- **Termination:**
  
  \( D(N,M) \) is distance
The Edit Distance Table

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

\[ D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases} \]
# EXECUTION

## The Edit Distance Table

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>O</th>
<th>I</th>
<th>T</th>
<th>N</th>
<th>E</th>
<th>T</th>
<th>N</th>
<th>I</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<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>
</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>
</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>
</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>
</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>
</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>
</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>
</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>
</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>
</tr>
<tr>
<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>
<td></td>
</tr>
</tbody>
</table>
Computing alignments

• Edit distance isn’t sufficient
  • We often need to **align** each character of the two strings to each other
• We do this by keeping a “backtrace”
• Every time we enter a cell, remember where we came from
• When we reach the end,
  • Trace back the path from the upper right corner to read off the alignment
# MinEdit with Backtrace

<table>
<thead>
<tr>
<th>n</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>8</td>
</tr>
<tr>
<td>i</td>
<td>7</td>
</tr>
<tr>
<td>t</td>
<td>6</td>
</tr>
<tr>
<td>n</td>
<td>5</td>
</tr>
<tr>
<td>e</td>
<td>4</td>
</tr>
<tr>
<td>t</td>
<td>3</td>
</tr>
<tr>
<td>n</td>
<td>2</td>
</tr>
<tr>
<td>i</td>
<td>1</td>
</tr>
<tr>
<td>#</td>
<td>0</td>
</tr>
<tr>
<td>#</td>
<td>e</td>
</tr>
<tr>
<td>x</td>
<td>e</td>
</tr>
<tr>
<td>c</td>
<td>u</td>
</tr>
<tr>
<td>t</td>
<td>i</td>
</tr>
<tr>
<td>o</td>
<td>n</td>
</tr>
</tbody>
</table>

The table shows the MinEdit with Backtrace algorithm for a given string. Each cell represents a character and its corresponding edit distance and backtrace directions. The shaded cells indicate the final backtrace for each character.
Adding Backtrace to Minimum Edit Distance

- **Base conditions:**
  \[
  D(i,0) = i \quad D(0,j) = j
  \]

- **Recurrence Relation:**
  For each \( i = 1..M \)
  For each \( j = 1..N \)
  \[
  D(i,j) = \min \begin{cases} 
  D(i-1,j) + 1 & \text{deletion} \\
  D(i,j-1) + 1 & \text{insertion} \\
  D(i-1,j-1) + 2; \text{if } X(i) \neq Y(j) \\
  0; \text{if } X(i) = Y(j) & \text{substitution}
  \end{cases}
  \]

- **Termination:**
  \( D(N,M) \) is distance

**Left, Down, Diagonal:**
- **Left:** \( \text{deletion} \)
- **Down:** \( \text{insertion} \)
- **Diagonal:** \( \text{substitution} \)
Weighted Edit Distance

• Why would we add weights to the computation?
  • Spell Correction: some letters are more likely to be mistyped than others
  • Biology: certain kinds of deletions or insertions are more likely than others
Confusion matrix for spelling errors

| X | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
| a | 0 | 0 | 7 | 1 | 342 | 0 | 0 | 2 | 118 | 0 | 1 | 0 | 0 | 3 | 76 | 0 | 0 | 1 | 35 | 9 | 9 | 0 | 1 | 0 | 5 | 0 |
| b | 0 | 0 | 9 | 9 | 2 | 2 | 3 | 1 | 0 | 0 | 0 | 0 | 5 | 11 | 5 | 0 | 10 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 8 | 0 | 0 |
| c | 6 | 5 | 0 | 16 | 0 | 9 | 5 | 0 | 0 | 0 | 1 | 0 | 7 | 9 | 1 | 10 | 2 | 5 | 39 | 40 | 1 | 3 | 7 | 1 | 1 | 0 |
| d | 1 | 10 | 13 | 0 | 12 | 0 | 5 | 5 | 0 | 0 | 2 | 3 | 7 | 3 | 0 | 1 | 0 | 43 | 30 | 22 | 0 | 0 | 4 | 0 | 2 | 0 |
| e | 388 | 0 | 3 | 11 | 0 | 2 | 2 | 0 | 89 | 0 | 0 | 3 | 0 | 5 | 93 | 0 | 0 | 14 | 12 | 6 | 15 | 0 | 1 | 0 | 18 | 0 |
| f | 0 | 15 | 0 | 3 | 1 | 0 | 5 | 2 | 0 | 0 | 0 | 3 | 4 | 1 | 0 | 0 | 0 | 6 | 4 | 12 | 0 | 0 | 2 | 0 | 0 | 0 |
| g | 4 | 1 | 11 | 11 | 9 | 2 | 0 | 0 | 0 | 1 | 1 | 3 | 0 | 0 | 2 | 1 | 3 | 5 | 13 | 21 | 0 | 0 | 1 | 0 | 3 | 0 |
| h | 1 | 8 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 12 | 14 | 2 | 3 | 0 | 3 | 1 | 11 | 0 | 0 | 2 | 0 | 0 | 0 |
| i | 103 | 0 | 0 | 0 | 146 | 0 | 1 | 0 | 0 | 0 | 6 | 0 | 0 | 49 | 0 | 0 | 0 | 2 | 1 | 47 | 0 | 2 | 1 | 15 | 0 |
| j | 0 | 11 | 1 | 9 | 0 | 0 | 1 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| k | 1 | 2 | 8 | 4 | 1 | 1 | 2 | 5 | 0 | 0 | 0 | 5 | 5 | 0 | 2 | 0 | 0 | 6 | 0 | 0 | 0 | 4 | 0 | 0 | 3 |
| l | 2 | 10 | 1 | 4 | 0 | 4 | 5 | 6 | 13 | 0 | 1 | 0 | 0 | 14 | 2 | 5 | 0 | 11 | 10 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| m | 1 | 3 | 7 | 8 | 0 | 2 | 0 | 6 | 0 | 0 | 4 | 4 | 0 | 180 | 0 | 6 | 0 | 0 | 9 | 15 | 13 | 3 | 2 | 2 | 3 | 0 |
| n | 2 | 7 | 6 | 5 | 3 | 0 | 1 | 19 | 1 | 0 | 4 | 35 | 78 | 0 | 0 | 7 | 0 | 28 | 5 | 7 | 0 | 0 | 1 | 2 | 0 | 2 |
| o | 91 | 1 | 1 | 13 | 116 | 0 | 0 | 0 | 25 | 0 | 2 | 0 | 0 | 2 | 14 | 0 | 0 | 2 | 4 | 14 | 39 | 0 | 0 | 0 | 18 | 0 |
| p | 0 | 11 | 1 | 2 | 0 | 6 | 5 | 0 | 2 | 9 | 0 | 2 | 7 | 6 | 15 | 0 | 0 | 1 | 3 | 6 | 0 | 4 | 1 | 0 | 0 | 0 |
| q | 0 | 0 | 1 | 0 | 0 | 0 | 27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| r | 0 | 14 | 0 | 30 | 0 | 0 | 12 | 2 | 0 | 2 | 8 | 2 | 0 | 5 | 8 | 4 | 20 | 1 | 14 | 0 | 0 | 12 | 22 | 4 | 0 | 0 | 1 | 0 | 0 |
| s | 11 | 8 | 27 | 33 | 35 | 4 | 0 | 1 | 0 | 1 | 0 | 27 | 0 | 6 | 1 | 7 | 0 | 14 | 0 | 15 | 0 | 0 | 5 | 3 | 20 | 1 |
| t | 3 | 4 | 9 | 42 | 7 | 5 | 19 | 5 | 0 | 1 | 0 | 14 | 9 | 5 | 5 | 6 | 0 | 11 | 37 | 0 | 0 | 2 | 19 | 0 | 7 | 6 |
| u | 20 | 0 | 0 | 0 | 44 | 0 | 0 | 0 | 64 | 0 | 0 | 0 | 0 | 2 | 43 | 0 | 0 | 4 | 0 | 0 | 0 | 2 | 0 | 8 | 0 | 0 |
| v | 0 | 0 | 7 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 8 | 3 | 0 | 0 | 0 | 0 |
| w | 2 | 2 | 1 | 0 | 1 | 0 | 0 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 |
| x | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| y | 0 | 0 | 2 | 0 | 15 | 0 | 1 | 7 | 15 | 0 | 0 | 0 | 2 | 0 | 6 | 1 | 0 | 7 | 36 | 8 | 5 | 0 | 0 | 1 | 0 | 0 |
| z | 0 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 5 | 0 | 0 | 0 | 2 | 21 | 3 | 0 | 0 | 0 | 0 | 3 | 0 |
Weighted Min Edit Distance

• Initialization:
  \[
  D(0,0) = 0 \\
  D(i,0) = D(i-1,0) + del[x(i)]; \quad 1 \leq i \leq N \\
  D(0,j) = D(0,j-1) + ins[y(j)]; \quad 1 \leq j \leq M 
  \]

• Recurrence Relation:
  \[
  D(i,j) = \min \left\{ \\
  D(i-1,j) + del[x(i)] \\
  D(i,j-1) + ins[y(j)] \\
  D(i-1,j-1) + sub[x(i),y(j)] \\
  \right\} 
  \]

• Termination:
  \[ D(N,M) \text{ is distance} \]
Alignments in two fields

• In Natural Language Processing
  • We generally talk about distance (minimized)
    • And weights

• In Computational Biology
  • We generally talk about similarity (maximized)
    • And scores