Natural Language Processing

Lecture 5—9/8/2015

Jim Martin

Today

- Finish up segmentation/tokenization
- HW discussion
- Evaluation issues
- Minimum edit distance
  - Dynamic programming

Issues in Tokenization

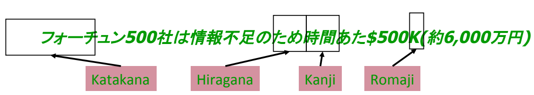
- Finland's capital → Finland Finlands Finland's ?
- what're, I'm, isn't → What are, I am, is not
- Hewlett-Packard → Hewlett Packard ?
- state-of-the-art → state of the art ?
- Lowercase → lower-case lowercase lower case ?
- San Francisco → one token or two?
- m.p.g., PhD. → ??

Tokenization: language issues

- French
  - ***L'ensemble*** → one token or two?
    - ***L*** ? ***L'***? ***Le*** ?
- German noun compounds are not segmented
  - ***Lebensversicherungsgesellschaftsangestellter***
  - 'life insurance company employee'
  - German information retrieval needs **compound splitter**

Tokenization: language issues

- Chinese has 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 Florid
- Japanese allows multiple alphabets intermingled



Case folding

- Applications like web search reduce all letters to lower case
  - Since users tend to use lower case
- For sentiment analysis, MT, Information extraction
  - Case is helpful (***US*** versus ***us; IRE vs. ire)***

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

Stemming

- Reduce terms to their stems
- *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 exampl compress and compress ar both accept as equival to compress

Porter's Algorithm

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
…
```

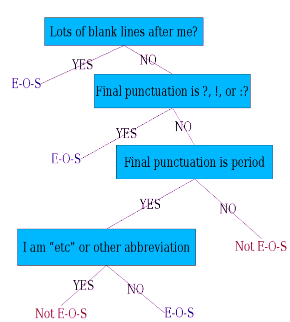
Complex Morphology

- Some languages require complex morpheme segmentation
  - Turkish
  - Uygarlastiramadiklarimizdanmissinizcasina
  - `(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'

Sentence Segmentation

- In English, puncuation is used to mark sentence boundraies
  - !, ? are relatively unambiguous
  - Period "." is quite ambiguous
  - Abbreviations like Inc. or Dr.
  - Numbers like .02% or 4.3
- Machine learning approach
  - Build a binary classifier
  - Looks at each possible EOS puncuation and decides EndOfSentence/NotEndOfSentence

Decision Tree Example



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)

Decision Trees and other classifiers

- We can think of the questions in a decision tree as features that could be exploited by any kind of classifier
  - Logistic regression
  - SVM
  - Neural Nets
  - etc.

Homework Preview

- English hashtag segmentation
  - #deflategate → deflate gate
- Using MaxMatch
  1. Implement it in python
  2. Evaluate how well it works
     - First make sure it does work
  3. Figure out how to improve it
     - (non-probablistically)

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

Maximum Matching

thematian

Testing, Improvement and Evaluation

- For this HW you have two tasks
  - Make sure that you have implemented MaxMatch correctly
    - Testing
  - Figure out a way to improve performance on this task
    - This means you need a way to detect improvement

Testing

- Given a particular dictionary and a correct implementation there is a "right" answer that MaxMatch should come up with. So for "themartian" that might be
  - them art i an
- That's the "right" answer even though its clearly not the right answer…

Improvement and Evaluation

- So given a test set of outputs like
  - them art i an
- How do we know if things are getting better?
- What do we need to know to say things are getting better?

Reference Answers

- We need to know what the "right" answer is. Here "right" means the answer we would expect a human to produce. In this case "the martian". So we have
  - them art i an
  - the martian
- And a whole bunch of examples like this, some right, some wrong
- How do we assess how well we're doing?

Evaluation

- Strict accuracy
  - Given a test set, how many things are right and how many are wrong?
- Too pessimistic
  - Might get you fired
- Not fine-grained enough
  - May not show you are making progress when you are in fact making progress

Progress?

- Start with
  - them art I an
  - the martian
- Move to
  - The mart I an
  - The martian
- They're both still wrong. So does it make sense to say one is better?

Edit Distance

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

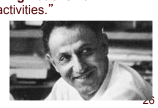
Note

- The following discussion has 2 goals
  1. Learn the minimum edit distance computation and algorithm
     1. To use in the HW
  2. Introduce dynamic programming

Why "Dynamic Programming"

"Where did the name, dynamic programming, come from? **The 1950s were not good years for mathematical research. We had a very interesting gentleman in Washington named Wilson. He was Secretary of Defense, and he actually had a pathological fear and hatred of the word, research**. I'm not using the term lightly; I'm using it precisely. His face would suffuse, he would turn red, and he would get violent if people used the term, research, in his presence. You can imagine how he felt, then, about the term, mathematical. The RAND Corporation was employed by the Air Force, and the Air Force had Wilson as its boss, essentially. Hence, I felt I had to do something to shield Wilson and the Air Force from the fact that I was really doing mathematics inside the RAND Corporation. What title, what name, could I choose? In the first place I was interested in planning, in decision making, in thinking. But planning, is not a good word for various reasons. **I decided therefore to use the word, "programming" I wanted to get across the idea that this was dynamic, this was multistage, this was time-varying I thought, lets kill two birds with one stone. Lets take a word that has an absolutely precise meaning, namely dynamic, in the classical physical sense. It also has a very interesting property as an adjective, and that is its impossible to use the word, dynamic, in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. Its impossible. Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to**. So I used it as an umbrella for my activities."

Richard Bellman, "Eye of the Hurricane: an autobiography" 1984.



Min Edit Example

	i n t e n t i o n
delete i →	n t e n t i o n
substitute n by e →	e t e n t i o n
substitute t by x →	e x e n t i o n
insert u →	e x e n u t i o n
substitute n by c →	e x e c u t i o n

Minimum Edit Distance

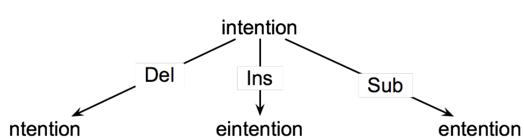
```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
d s s   i s
```

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

Min Edit As Search

- That's all well and good but how did we find that particular (minimum) set of operations for those two strings?
- We can view edit distance as a search for a path (a sequence of edits) that gets us from the start string to the final string
  - 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: the number of edits

Min Edit as Search



Min Edit As Search

- But that generates a huge search space
- Navigating that space in a naïve backtracking fashion would be incredibly wasteful
- Why?

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

Defining Min Edit Distance

- For two strings S_1 of len *n*, S_2 of len *m*
  - distance(*i,j*) or D(*i,j*)
    - Is the min edit distance of $S_1[1..i]$ and $S_2[1..j]$
      - That is, the minimum number of edit operations need to transform the first *i* characters of S_1 into the first *j* characters of S_2
    - The edit distance of S_1, S_2 is D(*n,m*)
- We compute D(*n,m*) by computing D(*i,j*) for all *i* (0 < *i* < n) and *j* (0 < j < m)

Defining Min Edit Distance

- Base conditions:
  - D(*i*,0) = *i*
  - D(0,*j*) = *j*
  - Recurrence Relation*:*
  - $$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

Dynamic Programming

- A tabular computation of D(n,m)
- Bottom-up
  - We compute D(i,j) for small i,j
  - And compute larger D(i,j) based on previously computed smaller values

The Edit Distance Table

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

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

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

Min Edit Distance

- Note that the result isn't all that informative
  - For a pair of strings we get back a single number
    - The min number of edits to get from here to there
- That's like a map routing program that tells you the distance from here to Denver but doesn't tell you how to get there.

Paths

- Keep a back pointer
  - Every time we fill a cell add a pointer back to the cell that was used to create it (the min cell that lead to it)
  - To get the sequence of operations follow the backpointer from the final cell

Adding Backtrace to MinEdit

- Base conditions:
  - $D(i,0) = i$
  - $D(0,j) = j$
- Recurrence Relation:
  - $D(i,j) = \min \begin{cases} D(i-1,j) + 1 & \text{Case 1} \\ D(i,j-1) + 1 & \text{Case 2} \\ D(i-1,j-1) + \begin{cases} 1; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} & \text{Case 3} \end{cases}$

$$ptr(i,j) \begin{cases} \text{LEFT} & \text{Case 1} \\ \text{DOWN} & \text{Case 2} \\ \text{DIAG} & \text{Case 3} \end{cases}$$

Complexity

- Time:

  O(nm)
- Space:

  O(nm)
- Backtrace

  O(n+m)

Alignments

- An alignment is a 1 to 1 pairing of each element in a sequence with a corresponding element in the other sequence or with a gap...

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

```
-AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---
TAG-CTATCAC--GACCGC--GGTCGATTTGCCCGAC
```

Weighted Edit Distance

- Why would we want to add weights to the minimum edit distance computation?

Confusion matrix

sub[X, Y] = Substitution of X (incorrect) for Y (correct)

X \ Y (correct)	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	5	11	5	0	10	0	0	2	1	0	0	8	0	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	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	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	1	1	9	0	0	1	0	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	0	5	0	2	0	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	3	116	0	0	0	25	0	2	0	0	0	0	14	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	0
r	0	14	0	30	12	2	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	0	2	0	8	0
v	0	0	7	0	0	3	0	0	0	0	0	1	0	0	1	0	0	0	8	3	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	0
x	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	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	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0

DP Search

- In the context of language processing (and signal processing) this kind of algorithm is often referred to as a DP search
  - Min edit distance
  - Viterbi and Forward algorithms
  - CKY and Earley
  - MT decoding

Back to the HW

- How to measure improvement
  - Length normalized minimum edit distance
    - AKA: word error rate.
      - Minimum edit distance/length of reference answer averaged over the development/test corpus
- What kind of improvement?
  - The lexicon
    - Starting with 75k words from google
  - The heuristic
    - Longest match
  - The search
    - Greedy

Next Time

- Language modeling
  - Read the new draft Chapter 4