CS4025: Words

- Words
- Bayesian Reasoning with words:
  - Spelling correction
  - Sentiment Analysis

Reading: J&M (5.1 to 5.6 in 1st ed, 3.10 to 3.11 in 2nd)

Spelling correction

Fred's dog likes to chase our cag.

- What is the misspelled word?
- What should it be?

Spelling errors

- 80% of all spelling errors are
  - character insertion (*the* -> *ther*)
  - character deletion (*the* -> *th*)
  - character substitution (*the* -> *thw*)
  - character transposition (*the* -> *teh*)
- This defines a set of candidates
  - *cag* <- *car*, *cat*, *cage*, *cga*, ...

Example: *acress*

Which words in the dictionary can be obtained by a single transformation?

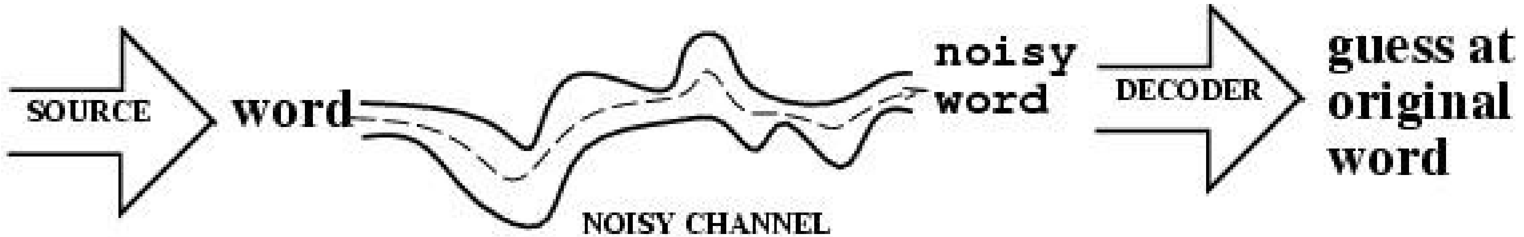
		Transformation			
Error	Correction	Correct Letter	Error Letter	Position (Letter #)	Type
acress	actress	t	–	2	deletion
acress	cress	–	a	0	insertion
acress	caress	ca	ac	0	transposition
acress	access	c	r	2	substitution
acress	across	o	e	3	substitution
acress	acres	–	2	5	insertion
acress	acres	–	2	4	insertion

How choose between these?

- Statistical model of error frequency
  - » e->r more common than e -> p (keyboard)
- Statistical model of words
  - » *cat* is more common than *cage*
- AI: world knowledge, what's plausible
  - » dogs like to chase cats

A combined statistical model

The noisy channel model:



What comes out is affected by what went in and how the channel distorts it.

The model precisely

- Assume we have received the noisy word O
- We seek to choose the possible input w which maximises P(w|O)
- P(w|O) is the probability that w was intended, given that O was received
- By Bayes' rule, this is equivalent to

$$P(w|O) = \frac{P(O|w) \cdot P(w)}{P(O)}$$

The model precisely - 2

- We know O, and need to find the value of w that maximises:

$$P(w|O) = \frac{P(O|w) \cdot P(w)}{P(O)}$$

- Since P(O) is the same for all w, we just need to maximise:

$$P(O|w) \cdot P(w)$$

- P(w) is the prior (depends on the *language* model)
  - Predict more likely words
- P(O|w) is the conditional likelihood (depends on the *channel* model)
  - Predict more likely mis-spellings

The language model

- How do we estimate P(w) for a possible word w?
- We collect a large corpus of text and see what proportion of words are w:

$$\frac{Number\ of\ times\ w\ appears}{Total\ number\ of\ words}$$

- In practice, we need to "smooth" the counts to cater for the fact that a possible word may not appear in the corpus (see J&M)

The channel model

- How likely are the various transformations of the original word?

Option 1: Use a corpus of errors to estimate types of errors (Birkbeck spelling error corpus http://ota.ahds.ac.uk/texts/0643.html)

- Count e.g.
  - trans[x,y] is how often the characters xy were typed as yx
  - count[x,y] is how often xy occurs in the correct words
- If O is the result of transposing x and y:
  - P(O|w) = trans [x,y] / count [x,y]

The channel model

- How likely are the various transformations of the original word?

Option 2: Use your knowledge of typing and language to create the channel model.

- Assign higher probability to words which require fewer transforms
- Vowel substitution is more common than consonant substitution
- Making a mistake on the first letter is less likely
- Errors between adjacent letters on keyboard are more likely
- Insertion errors are more likely if the same character is repeated
- …..

Example

c	freq(c)	p(c)	p(t|c)	p(t|c)p(c)	%
actress	1343	.0000315	.000117	3.69×10^{-9}	37%
cress	0	.00000014	.00000144	2.02×10^{-14}	0%
caress	4	.0000001	.00000164	1.64×10^{-13}	0%
access	2280	.000058	.000000209	1.21×10^{-11}	0%
across	8436	.00019	.0000093	1.77×10^{-9}	18%
acres	2879	.000065	.0000321	2.09×10^{-9}	21%
acres	2879	.000065	.0000342	2.22×10^{-9}	23%

- % is the result of dividing by P(acress) and multiplying by 100
- Note that *acres* can be obtained in 2 ways and so will win overall.

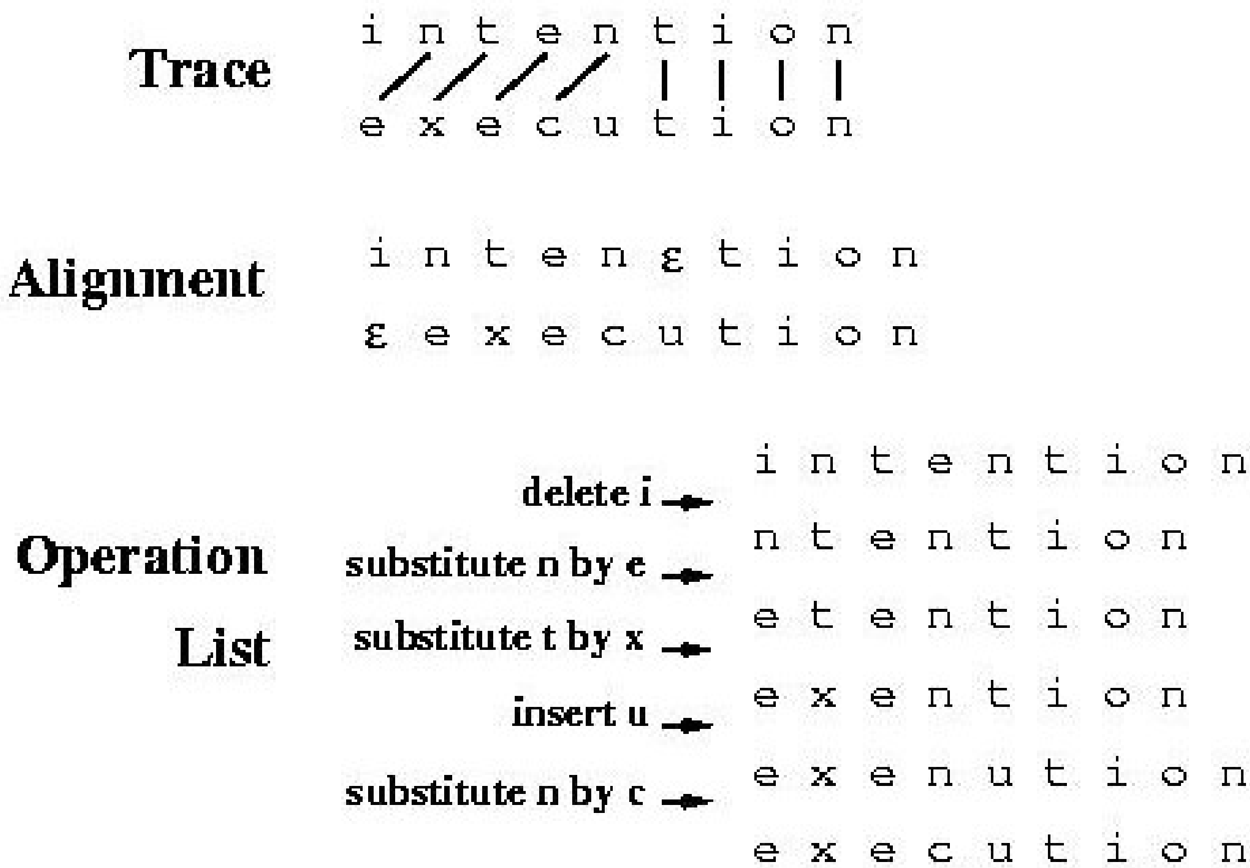
Algorithm 1 for single errors

```
function SpellingChecker (Dictionary, Text)
    1) Load Dictionary (as hash table usually)
    2) For each word in text
        (a) If word in Dictionary
            (i) do nothing
         Else
            (i) apply noisy channel model to correct
```

Dealing with multiple errors

- If we assume errors are independent, if there are several we can just multiply their probabilities.
- But if the words are very different, it may be hard to align them and determine the set of errors made.
- We need to find the minimum set of operations needed to transform one word into another – the *minimum edit distance*. Then we can compute P(O|w) using this set.
- There are different ways to visualise such alignments:

Dealing with multiple errors



Finding the minimal edit distance

- Assume that each possible edit has a cost, e.g. insertions and deletions cost 1, substitutions have a cost of 2. (Costs can depend on the characters).
- E.g. the alignment on the last slide scores 8. Is this the best alignment?
- Maintain an array *edit-distance* such that:
  - » *edit-distance[i,j]* = the minimal distance (sum of costs) between the first i characters of the target and the first j characters of the source.
- Each cell can be computed as a function of the surrounding cells:

Example

Source

n	9	10	11	10	11	12	11	10	9	**8**
o	8	9	10	9	10	11	10	9	**8**	9
i	7	8	9	8	9	10	9	**8**	9	10
t	6	7	8	7	8	9	**8**	9	10	11
n	5	6	7	6	7	**8**	9	10	11	10
e	4	3	6	**5**	**6**	7	8	9	10	11
t	3	4	**5**	6	7	8	7	10	11	12
n	2	3	**4**	5	6	7	8	8	10	9
i	1	**2**	3	4	5	6	7	6	9	10
#	**0**	1	2	3	4	5	6	7	8	9
	#	e	x	e	c	u	t	i	o	n

Target

The idea

- An alignment is a path from [0,0] to [m,n] through adjacent cells. $\nearrow$ = subst, $\uparrow$ = del, $\longrightarrow$ = insert
- Best path through cell “?” could involve:

X	?
Y	Z

Passing through Y and then substituting
Passing through X and then inserting, or
Passing through Z and then deleting

- Best score for this cell is **min(Y+subst, X+ins, Z+del)** where ins and del are 1, subst is 0 or 2, depending on whether the characters are the same or not

The algorithm (sketch)

```
function min-edit-distance(target,source)-> min-distance
  n = length(target); m = length(source);
  create a distance matrix distance[n+1,m+1];
  distance[0,0] = 0;
  for all i, distance[i,0] = ins-cost * i;
  for all j, distance[0,j] = del-cost * j;
  for each column i from 1 to n do
     for each row j from 1 to m do
        distance[i,j] =
          min(distance[i-1,j] + ins-cost,
          distance[i,j-1] + del-cost,
          distance[i-1,j-1]
                     + subst-cost(source[j],target[i]))
  return distance[n,m];
```

Example

Source

n	9	10	11	10	11	12	11	10	9	**8**
o	8	9	10	9	10	11	10	9	**8**	9
i	7	8	9	8	9	10	9	**8**	9	10
t	6	7	8	7	8	9	**8**	9	10	11
n	5	6	7	6	7	**8**	9	10	11	10
e	4	3	6	**5**	**6**	7	8	9	10	11
t	3	4	**5**	6	7	8	7	10	11	12
n	2	3	**4**	5	6	7	8	8	10	9
i	1	**2**	3	4	5	6	7	6	9	10
#	**0**	1	2	3	4	5	6	7	8	9
	#	e	x	e	c	u	t	i	o	n

Target

Reading off the solution

n	9	10	11	10	11	12	11	10	9	**8**
o	8	9	10	9	10	11	10	9	**8**	9
i	7	8	9	8	9	10	9	**8**	9	10
t	6	7	8	7	8	9	**8**	9	10	11
n	5	6	7	6	7	**8**	9	10	11	10
e	4	3	6	**5**	**6**	7	8	9	10	11
t	3	4	**5**	6	7	8	7	10	11	12
n	2	3	**4**	5	6	7	8	8	10	9
i	1	**2**	3	4	5	6	7	6	9	10
#	**0**	1	2	3	4	5	6	7	8	9
	#	e	x	e	c	u	t	i	o	n

Real-world errors

- What if a word is misspelled as another word?
  - Typographical errors
    - insertion, deletion, substitution and transposition
    - *buckled* for *bucked*, *his* for *this*
  - Homophones
    - *dessert* for *desert*, *piece* for *peace*

Fred's dog likes to chase our chat

- Legal, but not very plausible
  - more likely *chat* is misspelling of *cat*
- How detect this? Need to assess plausibility of the sentence
  - Calculate Likelihood of sequences of words (ngrams)

First Practical: Sentiment Analysis

TASK: Given a textual review of a movie, we need to decide if it is Positive or Negative.

- it's so laddish and juvenile , only teenage boys could possibly find it funny .
- take care of my cat offers a refreshingly different slice of asian cinema .
- interesting but not compelling
- everytime you think undercover brother has run out of steam , it finds a new way to surprise and amuse .

Bayesian Approach

To calculate the likelihood of a review being positive based on a single word:

- w is a single word in the review
- *P(positive|w) = P(w|positive) * P(positive) / P(w)*

P(positive) is fraction of reviews that are positive in the collection of reviews (prior likelihood of a review being positive)

P(w) is the prior likelihood of seeing w

- *P(w) = freq. of w in all reviews/total number of words in all reviews*

P(w|positive) the chance of seeing w in a positive review

P(w|positive) = count of w in positive reviews / total words in positive reviews

Naive Bayes Algorithm

- This is the simplest machine learning Algorithm
  - Assumes that all features (words) are independent.

$P(Pos|w1,w2,.....,wn) = P(w1,w2..wn|Pos)*P(Pos) / P(w1,w2...wn)$

$P(w1,w2..wn|Pos) = P(w1|Pos)* P(w2|Pos)....*P(wn|Pos)$

$P(w1,w2...wn) = P(w1)*P(w2)*....*P(wn)$

To simplify, calculate:

$Score(Pos) = P(Pos) * P(w1|Pos)* P(w2|Pos)*....*P(wn|Pos)$

$Score(Neg) = P(Neg) * P(w1|Neg)* P(w2|Neg)*....*P(wn|Neg)$

Return the sentiment with higher score

Naive Bayes Probability Distribution

*Score(Pos) = P(Pos) * P(w1|Pos)* P(w2|Pos)*....*P(wn|Pos)*

*Score(Neg) = P(Neg) * P(w1|Neg)* P(w2|Neg)*....*P(wn|Neg)*

If you want a probability distribution (to determine confidence of classification, for example)

P(Pos|w1..wn) = Score(Pos,w1..wn) / (Score(Pos,w1..wn) + Score(Neg,w1..wn))

P(Neg|w1..wn) = Score(Neg,w1..wn) / (Score(Pos,w1..wn) + Score(Neg,w1..wn))

Naive Bayes Probability Distribution

Naive Bayes can be used for any text classification task,

For example: email spam filter:

*Score(Spam) = P(Spam) * P(w1|Spam)* P(w2|Spam)*....*P(wn|Spam)*

*Score(Nospam) = P(Nospam) * P(w1|Nospam)* P(w2|Nospam)*....*P(wn|Nospam)*

What if you have more than two classes? Positive, Negative and Neutral