University of Malta
BSc(IT) HonsYear IV

CSA4050: Advanced Topics in NLP

Statistical NLP IV

Spelling Correction

- Spelling Correction
- Noisy Channel Method
- Probabilistic Models
- Bayesian Method

Dept Computer Science and AI 2003/04
Lecturer: Michael Rosner

Acknowledgement

Much of the material for this lecture comes from chapter 5 of Daniel Jurafsky/Jim Martin: Speech and Language Processing.

www.cs.colorado.edu/∼martin/slp.html

See also

www.sultry.arts.su.edu.au/links/statnlp.html

for resources on statistical NLP

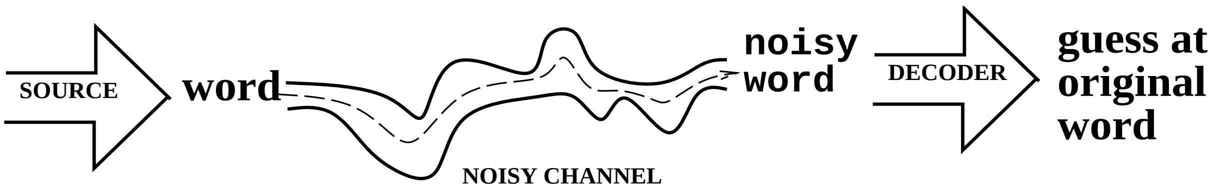
Speech Recognition and Spelling Correction

Despite apparent differences, these problems share many underlying similarities.

Both are concerned with the problem of accepting a string of symbols and mapping them to a sequence of progressively less likely words.

- In spelling correction, the symbols are characters.
- In the recognition of pronunciation variations, the symbols are *phones*.

Noisy Channel Model



- Language is generated and passed through a noisy channel.
- Resulting noisy data are received
- Goal is to recover the original data from the noisy data.
- Same model can be used in diverse areas of language processing e.g. spelling correction, morphological analysis, pronunciation modelling, machine translation.
- Metaphor comes from Jelinek's (1976) work on speech recognition, but algorithm is a special case of Bayesian inference (1763).

Spelling Correction

Kukich(1992) breaks the field down into three increasingly broader problems:

- Detection of non-words (e.g. *graffe*).
- Isolated word error correction (e.g. *graffe* $\Rightarrow$ *giraffe*).
- Context dependent error detection and correction where the error may result in a valid word (e.g. *there* $\Rightarrow$ *three*) .

Spelling Error Patterns

According to Damereau (1964) 80% of all misspelled words are caused by **single-error misspellings** which fall into the following categories (for the word *the*)

- Insertion (*ther*).
- Deletion (*th*).
- Substitution (*thw*).
- Transposition (*hte*).

Because of this study, much subsequent research focused on the correction of single error misspellings.

Causes of Spelling Errors

Keyboard Based

- Immediately adjacent keys in the same row of the keyboard (50% of the novice substitutions (31% of all substitutions).
- Hitting corresponding key on opposite side of keyboard.
- 83% novice and 51% overall were keyboard errors

Cognitive

- Phonetic *seperate* - *separate*
- Homonym *there* - *their*.

OCR: mainly visual similarity (e.g. *m* - *rn*; substitutions; space deletions or insertions; failures.

Bayesian Classification

In this task, we are given some observation and we must determine which of a set of classes it belongs to.

For example, in *speech recognition*

- The observation is a string of phones
- The classification is the word that was said

For example, in *spelling correction*

- The observation is a string of characters
- The classification is the word that was intended.

An Example

- We are given a string $O = (o_1, \ldots o_n)$ of *observations*.
- The Bayesian interpretation begins by considering all possible classes, i.e. the set of all possible words.
- Out of this universe, we want to choose that word w in the vocabulary V which is most probable given the observation that we have O ., i.e.

  ŵ =argmax$_{w \in V} P(w \mid O)$

  where argmax$_x f(x)$ means "the x such that f(x) is maximised".
- *Problem*. Whilst this is guaranteed to give us the optimal word, it is not obvious how to make the equation operational: for a given word w and a given O we don't know how to compute $P(w \mid O)$

Bayes' Rule

The intuition behind Bayesian classification is use Bayes' rule to transform $P(w \mid O)$ into a product of two probabilities, each of which is easier to compute than $P(w \mid O)$

Bayes' Rule

$P(x \mid y) = \frac{P(y|x)P(x)}{P(y)}$

we obtain

$\hat{w} = \text{argmax}_{w \in V} P(w \mid O) = \frac{P(O|w)P(w)}{P(O)}$

- We can estimate $P(w)$ from the frequency of the word.
- As we shall see, $P(O \mid w)$ is also easy to estimate.
- $P(O)$, the probability of the observation sequence, is harder to estimate, but we can ignore it.

Prior Probability and Likelihood

- **We can ignore** $P(O)$

  since we are maximising

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

  for all words where the denominator never changes. So ŵ, the most likely word

  $= \text{argmax}_{w \in V} \; \frac{P(O|w)P(w)}{P(O)}$

  $= \text{argmax}_{w \in V} \; P(O \mid w)P(w)$

  The two terms of this product have names:

  $P(w)$ is called the **prior probability**

  $P(O \mid w)$ is called the **likelihood**

- In case of spelling O is observed typo and w is correct word.

Bayesian Classification Applied To Spelling Correction

- The noisy channel approach was first suggested by Kernighan, Church and Gale (1990)
- Their program (called *correct*)
  - Takes words rejected by the Unix *spell* progam
  - Generates a list of potentially correct words
  - Ranks them according to the the above equation
  - Picks the one with the highest rank
- We will follow the correction of the word *acress* which proceeds in two steps, proposing candidates, and ranking candidates.

Proposing Candidates

- Assume that correct word will differ from the misspelling by a single insertion, deletion, substitution or transposition.
- The list of candidates is generated from the typo by applying any single transformation that results in a word occurring in a large online dictionary
- For instance, the typo *acress* yields the following list: actress (d), cress (i), caress (t), access (s), across (s), acres (i), acres (i)

		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

Ranking Candidates

The second stage scores each correction. Let t be the typo and c range over a set C of candidate corrections. The most likely correction ĉ is then $\text{argmax}_{c \in C} P(c \mid t)$, which by Bayes' rule is equivalent to

$$\hat{c} = \text{argmax}_{c \in C} P(t \mid c) P(c)$$

The prior probability P(c) can be estimated by

- Counting how often the word c appears in the corpus
- Normalising the count by dividing it by the number N of words in the corpus. Zero counts can cause problems, and so we add .5 to all counts (this is called "smoothing"). Having done this, we must compensate by adding 0.5*V to the denominator for each word V in the vocabulary so that

  $$P(c) = \frac{C(c)+0.5}{N+0.5V}$$

Prior Probability

Using the formula above, the prior probabilities come out as follows:

c	**freq(c)**	**P(c)**
actress	1343	.0000315
cress	0	.000000014
caress	4	.0000001
access	2280	.000058
across	8436	.00019
acres	2879	.000065

Computing the Likelihood Term $P(t \mid c)$

The likelihood term $P(t \mid c)$ is difficult if not impossible to *predict* in the general case (depends on arbitrary factors e.g. on who the typist is, the lighting conditions etc.)

However it can be *estimated* if we have a theory of how it was produced.

In the case of Kernighan et al, it is assumed that t arises from a single insertion, deletion, transposition or substitution, for which certain *a priori* probabilities are evident, e.g.

- the identity of the correct letter,
- how the letter was misspelled
- the surrounding context
- e.g. m and n are often confused (because they are pronounced similarly, they often crop up in the same contexts.

Computing $P(t \mid c)$

In fact Kernighan et al. ignored most of these factors and simply counted the occurrences of particular kinds of error occurring in a large corpus of errors.

Using this technique they constructed a series of **confusion matrices** for the different kinds of error.

- sub[x,y] - number of times x is substituted for y.
- ins[x,y] - number of times x was typed as xy
- del[x,y] - number of times xy was typed as x
- tran[x,y] - number of times xy was typed as yx

Using the Confusion Matrices

Using these matrices, they calculated $P(t \mid c)$ as follows, where c_p is the p^{th} character of word c.

$$P(t \mid c) = \frac{\text{sub}[t_p, c_p]}{\text{count}[c_p]}$$

$$P(t \mid c) = \frac{\text{tran}[c_p, c_{p+1}]}{\text{count}[c_p c_{p+1}]}$$

$$P(t \mid c) = \frac{\text{del}[c_{p-1}, c_p]}{\text{count}[c_{p-1}, c_p]}$$

$$P(t \mid c) = \frac{\text{ins}[c_{p-1}, t_p]}{\text{count}[c_{p-1}]}$$

Result

c	freq(c)	p(c)	p(t|c)	p(t|c)p(c)	%
actress	1343	.0000315	.000117	3.69×10^{-9}	**37%**
cress	0	.000000014	.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%**

- The algorithm predicts "acres" as the correct word. Yet the surrounding context

  ... was called a "stellar and versatile acress whose combination of sass and glamour has defined her..."

  makes it clear that the correct word is actually "actress".

- Clearly other methods are necessary to take account of context.

Producing Confusion Matrices

The algorithm described requires hand-annotated data to train the confusion matrices. This is expensive and slow to produce.

Kernighan et al (1990) suggested the following iterative approach to the problem of constructing confusion matrices:

- Initialise matrices with equal values
- Spelling error correction algorithm is run on a set of spelling errors. This yields the errors paired with their corrections.
- Using this information in these pairs, the confusion matrices can now be recomputed.
- The program performed quite well, agreeing with 87% of the judgements of asking human judges concerning spelling errors that had two possible corrections.