Engineering Part IIB: Module 4F11
Speech and Language Processing
Lecture 7 : Statistical Language Models

Phil Woodland: pcw@eng.cam.ac.uk

Lent 2014

Cambridge University Engineering Department

Statistical Speech Recognition

The aim of speech recognition is to find the word string, **W**, such that it maximises $P(\mathbf{W}|\mathbf{O})$ where $\mathbf{O}$ is the observed acoustic data.

$$P(\mathbf{W}|\mathbf{O}) = \frac{p(\mathbf{O}|\mathbf{W})P(\mathbf{W})}{p(\mathbf{O})}$$

$p(\mathbf{O}\|\mathbf{W})$	obtained from the acoustic model
$P(\mathbf{W})$	obtained from the language model (LM)
$p(\mathbf{O})$	normalising factor

$p(\mathbf{O})$ is independent of the word sequence, so does not affect the choice of most probable word sequence.

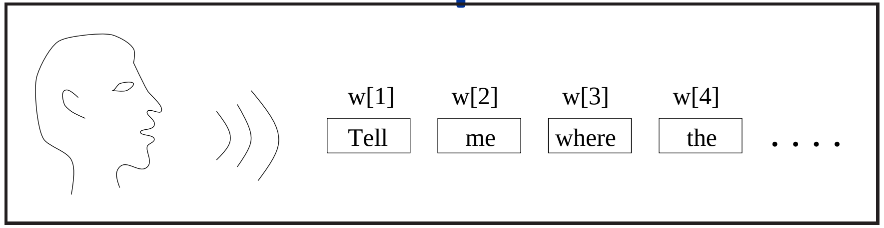
The LM should model:

- syntactic structure;
- semantic information;

of natural language. Language models are also used for optical character recognition, and machine translation (see later in course), ...

Language Modelling: General Principles

For the sequence of words:



$$\mathbf{W} = w(1), w(2), \ldots, w(k-1)$$

Terminology:

$w(k)$ will refer to the kth word in a sequence of words.

w_i will refer to the ith word in the vocabulary.

The LM computes probability of this word sequence, $P(\mathbf{W})$, (however unlikely). It should make speech recognition simpler by reducing the probability of highly unlikely word sequences.

The language model also gives a measure of task complexity. For example

- telephone numbers: a digit may be followed by any other digit (10);
- English: equivalent to *on average* followed by about 32 words.

This is related to *entropy*. Language models are more normally described by their perplexity (or average branching factor).

Perplexity

The perplexity (PP) is related to the entropy (H) of the language model

$$\mathrm{PP} = 2^H \quad \text{or} \quad H = \log_2 \mathrm{PP}$$

The probability of a word sequence is often decomposed into the product of word-prediction conditional probabilities:

$$P(w(1)w(2)\ldots w(M)) = \prod_{k=1}^{M} P\left(w(k)|w(1)\ldots w(k-1)\right)$$

The entropy of this sequence is (letting $M \to \infty$ to obtain a good estimate)

$$H = \lim_{M\to\infty} -\frac{1}{M}\sum_{k=1}^{M} \log_2 P(w(k)|w(1)\ldots w(k-1))$$

The **perplexity** is given by

$$
\begin{aligned}
PP &= \lim_{M\to\infty} \left(P(w(1)w(2)\ldots w(M))^{-\frac{1}{M}} \right) \\
&= \lim_{M\to\infty} \left(\prod_{k=1}^{M} P(w(k)|w(1)\ldots w(k-1)) \right)^{-\frac{1}{M}}
\end{aligned}
$$

In practice, will only have S sequences of words on which to estimate the entropy of the language model. The entropy is

$$
H = -\frac{1}{\sum_{s\in S} M_s} \left(\sum_{s\in S} \sum_{k=1}^{M_s} \log_2 P(w(k)|w(1)\ldots w(k-1)) \right)
$$

When we compute the perplexity using this over a corpus of test sentences, we get the **Test Set Perplexity** which is the value normally quoted.

Simple Example

If the probability of a word occurring is independent of all previous words then

$$P(w(k)|w(1)\dots w(k-1)) = P(w(k))$$

In this case, compute the perplexity directly from the LM (vocab size V) as

$$H = -\sum_{i=1}^{V} P(w_i)\log_2 P(w_i)$$

Note the limiting cases:

- all words equally likely: $H = -\sum_i \frac{1}{V}\log_2\frac{1}{V} = \log_2 V$ $\quad$ $\mathrm{PP} = 2^{\log_2 V} = V$
- only one word possible: $P(w_i) = \left\{ \begin{array}{c} 1, i = i' \\ 0, i \neq i' \end{array} \right\}$

  $H = 0$, hence PP$= 1$

Word Start/End Symbols

Two special symbols are usually added to the vocabulary

- sentence start symbol $<$s$>$
- sentence end symbol $<$/s$>$

These model the fact that sentences are of finite length, and that the position in a sentence is relevant for computing word probabilities.

The language model is thus trained on sequences of the form

$$<\texttt{s}>, w(1), w(2), \ldots, w(M), <\texttt{/s}>$$

- $P(<\texttt{s}>) = 0$ except at the start of each sentence, i.e. $w(0)$, when it is $= 1$
- $P(<\texttt{/s}>)$ can be used to compute the average sentence length, this is $w(M+1)$

$$\text{average sentence length} = \frac{1}{P(<\texttt{/s}>)}$$

N-Gram Language Models

The probability of a word sequence may be expressed as

$$P(w(1)\dots w(M+1)) = \prod_{k=1}^{M+1} P\left(w(k)|w(1)\dots w(k-1)\right)$$

The LM is required to estimate $P\left(w(k)|w(1)\dots w(k-1)\right)$ for any word sequence. For any reasonable size of vocabulary this is impractical to model directly.

It is usual to restrict the size of the **history** to the previous $N-1$ words. This is the N-**gram** language model. Thus

$$P(w(k)|w(1)\dots w(k-1)) \approx P(w(k)|w(k-N+1)\dots w(k-1))$$

Most frequently used are the unigram ($N=1$), bigram ($N=2$) and trigram ($N=3$), and 4-gram LMs. We need to make N as large as possible consistent with the ability to estimate the parameters from available training data.

N-Grams (cont)

For the trigram, $N = 3$.

$$P(w(k)|w(1)\ldots w(k-1)) \approx P(w(k)|w(k-2)w(k-1))$$

As with the acoustic model training the LM parameters are estimated using techniques based on *maximum likelihood* training.

The basic estimation is to use relative frequencies to estimate probabilities. Therefore to estimate the probability of a particular trigram it is necessary to find the "count" (frequency of occurrence) of triple $w_i w_j w_k$ in the training data and then

$$\hat{P}(w_k|w_i, w_j) = \frac{f(w_i, w_j, w_k)}{\sum_{k=1}^{V} f(w_i, w_j, w_k)} = \frac{f(w_i, w_j, w_k)}{f(w_i, w_j)}$$

where $f(a, b, c, \ldots) =$ number of times that the word sequence ($event$) "a b c . . . " occurs in the training data. This relative frequency approach is the ML estimate of the N-gram parameters.

N-Gram Advantages

N-grams are popular as

- they can be computed from real data
- they guarantee full coverage of all word sequences
- they simultaneously encode syntax, semantics and pragmatics
- they concentrate on very local dependencies
- they are very simple to compute during recognition - essentially a single table lookup
- they work!

Example N-Gram Generation

The longer the context of an N-gram, then at least in principle (subject to problems of parameter estimation), the more accurate is the model. This can be illustrated by generating example sentences from N-gram models (viewed as a Markov source). Jurafsky & Martin give some examples from N-Gram LMs trained on the complete works of Shakespeare (867k word tokens with punctuation treated as separate words):

Unigram :

Will rash been and by I the me loves gentle me not slavish page, the and hour; ill let

Bigram What means, sir. I confess she? then all sorts, he is trim, captain.

Trigram Sweet prince, Fallstaff shall die. Harry of Monmouth's grave.

4-gram Enter Leonato's brother Antonio, and the rest, but seek the weary beds of people sick.

Issues with N-Grams

The major issue is how to obtain **robust** estimates of the probabilities. For a vocabulary of V, there are V^N parameters to estimate. If $V = 640000$

- Trigram: $64000^3 = 2.62144 \times 10^{14}$ 4-gram: $64000^4 = 1.67772 \times 10^{19}$

The ML estimate is found from counts. Considering the trigram case:

- $f(w_i, w_j, w_k)$ will be zero for many word triples (& word-pairs). According to the language model this word sequence could *never* occur!

- if $f(w_i, w_j)$ is small, the ML estimate of $\hat{P}(w_i|w_j, w_k)$ will be unreliable.

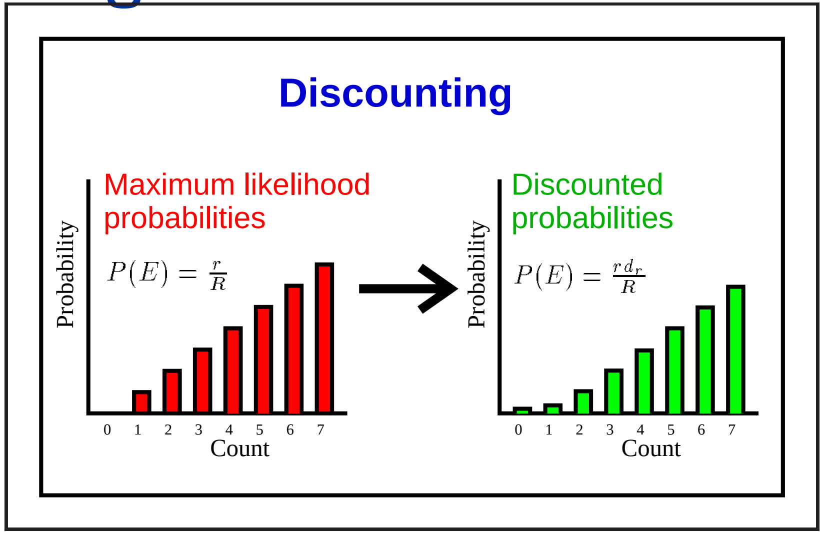
There are two (linked) aspects to current solutions:

a) find ways of estimating each N-gram probability using *discounting* to allow for events not seen in training.

b) use a more general distribution for unobserved N-grams *interpolation* or *backing-off*

Discounting

Assign some probability "mass" to unseen events. Probability estimates have sum-to-one constraints i.e. $\sum_{k=1}^{V} \hat{P}(w_k|w_i, w_j) = 1$. We can use relative-frequency based estimates if we **discount** (reduce) the counts of the seen events.



Therefore the N-gram estimate is modified to be

$$\hat{P}(w_k|w_i, w_j) = d(f(w_i, w_j, w_k)) \frac{f(w_i, w_j, w_k)}{f(w_i, w_j)}$$

where $d(r)$ is a *discount coefficient*. The amount by which the maximum likelihood estimate is altered depends on the frequency of the N-gram.

Example Discounting Schemes

- **Good Turing**: a popular form of discounting. This estimate assumes that the total probability (mass) for all events that did not occur in the training set is the same as the total observed probability (mass) for all events that occurred once. Hence $d(r)$ reduces the counts of the events that did occur so that a suitable amount of extra probability mass is available to distribute to the events that didn't occur in training. The overall effect is to employ a discounting formula

$$d(r) = \frac{(r+1)n_{r+1}}{r\ n_r}$$

where n_r is the number of N-grams occurring r times.

- **Absolute discounting**: here

$$d(r) = (r - b)/r$$

Typically $b = n_1/(n_1 + 2n_2)$. The discounting is applied to all counts. This is, of course, equivalent to simply subtracting the constant b from each count.

Backing Off

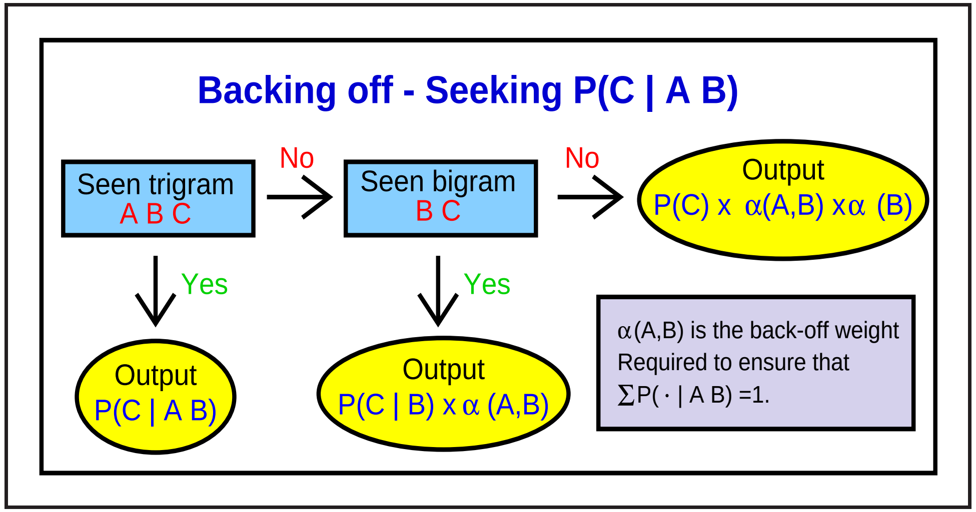
When N-grams cannot be estimated reliably because the count $f(w_i, w_j, \ldots)$ is too small, a more general distribution, normally based on an $(N-1)$-gram is used instead. E.g.

$$\hat{P}(w_j|w_i) = \begin{cases} d(f(w_i, w_j))\frac{f(w_i,w_j)}{f(w_i)} & \text{if } \; f(w_i, w_j) > C \\ \alpha(w_i)\hat{P}(w_j) & \text{otherwise} \end{cases}$$

$\alpha(w_i)$ is the *back-off* weight, it is chosen to ensure that

$$\sum_{j=1}^{V} \hat{P}(w_j|w_i) \;\; = \;\; 1$$

and C is the N-gram cut-off point (i.e. only N-grams that occur more frequently than this are retained in the final model). The value of C also controls the size of the resulting language model. For trigrams this results in the following form:



Here "Seen" means that the counts of the trigram, or bigram, exceed the cut-off point. This value may be set separately for the bigrams and trigrams.

Performance with varying N

Experiments on a broadcast news (BN) transcription task with a LM trained on 230MW (million words) of broadcast news transcriptions, newswire texts and acoustic transcriptions. Test on a BN data sets: BNeval97.

Recogniser uses an HTK state-clustered triphone HMM system (with unsupervised test-set adaptation) and a 65k word language model of various N. (HMMs trained with ML estimation with 140hrs data).

Lang Model Type	Perplexity BNeval97	% WER BNeval97
bigram	240	21.3
trigram	159	18.0
4-gram	147	17.3

- Most gain is from bigram to trigram
- Different test sets have different word error rates independent of perplexity
- PP change is a reasonable predictor of WER change e.g. a reduction of 39% in PP leads 19% in WER.

The results quoted above were generated in 1997. The best BN systems now have a WER less than 10% by using more advanced acoustic model parameter estimation and modelling techniques; more acoustic training data; and also well over a billion words of language model training data.

Interpolation

In backing off, the **longest history** that is felt to be reliable is used. Alternatively *all* N-grams can be smoothed together with appropriate weights.

Estimate the trigram by linear combination of trigram, bigram and unigram counts:

$$\hat{P}(w_k|w_i, w_j) = \lambda_3 \frac{f(w_i, w_j, w_k)}{f(w_i, w_j)} + \lambda_2 \frac{f(w_j, w_k)}{f(w_j)} + \lambda_1 \frac{f(w_k)}{\sum_{i=1}^{V} f(w_i)}$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$ to ensure a valid distribution.

- if trigram is rare, mainly use bigram and unigram frequencies i.e. $\lambda_1 > \lambda_2 > \lambda_3$
- if trigram is common, mainly use trigram frequencies i.e. $\lambda_3 > \lambda_2 > \lambda_1$

The λ values will normally be shared for example over e.g. all trigrams with particular count ranges.

If ML estimation is used to get the λ's the most complex N-gram would always be chosen (think about it). Instead *deleted interpolation* is commonly used.

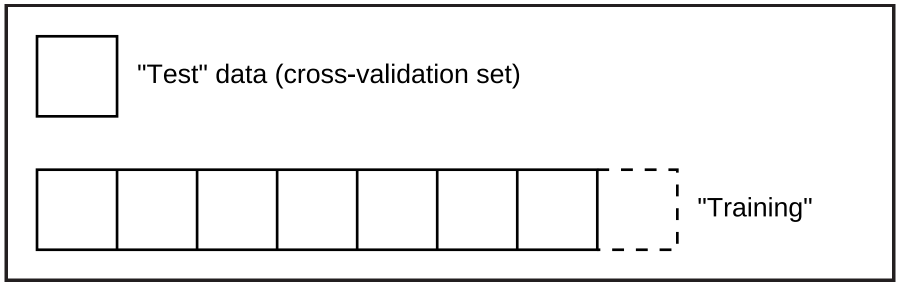
Deleted Interpolation

Divide the data into blocks and choose λ values to maximise the likelihood of deleted (or held-out) blocks of data. E.g. if the training data is divided into just 2 parts

1. (a) $\boxed{\text{Data 1}} \longrightarrow \hat{P}(w_k|w_i, w_j)$ in terms of unknown λ's
2. (b) $\boxed{\text{Data 2}} \longrightarrow P(\text{Data 2}) = \prod_i \hat{P}(w(i)|w(i-2), w(i-1))$
3. (c) Choose λ's to maximise log P(Data 2)

The key idea is that the λ values are chosen to maximise ability to predict unseen data. However, above method wastes data - so divide data into several blocks and rotate a the deleted block.

Now maximise $\sum_{i=1}^{M} \log P(\text{Data i})$
It can be shown that $\log P(\text{Data i})$ is *convex* with respect to the unknown parameters λ.



Summary

- Language Models are a key component in large vocabulary speech recognition systems
- Reduce the equivalent average number of word choices by several orders of magnitude
- Normally simple models based on N-**grams** are used
- Base prediction on previous N-1 words. Typically $N = 1 \ldots 4$.
- Need to deal with data **sparseness**: most N-grams don't occur in training data
- Two methods discussed:
  1. Use a combination of **discounting** and **backoff**
  2. Interpolation between different N-gram orders
- Can control N-gram size with the amount of back-off used.