# Empirical Methods in Natural Language Processing
# Lecture 3
# Language Modeling (I): From counts to smoothing

Philipp Koehn

14 January 2008

# Language models

- **Language models** answer the question: How likely is a string of English words good English?

  - *the house is big* → good
  - *the house is xxl* → worse
  - *house big is the* → bad

- Uses of language models

  - Speech recognition
  - Machine translation
  - Optical character recognition
  - Handwriting recognition
  - Language detection (English or Finnish?)

School of **informatics**

# Applying the chain rule

- Given: a string of English words $W = w_1, w_2, w_3, ..., w_n$

- Question: what is $p(W)$?

- Sparse data: Many good English sentences will not have been seen before.

$\rightarrow$ Decomposing $p(W)$ using the chain rule:

$$p(w_1, w_2, w_3, ..., w_n) = p(w_1) \, p(w_2|w_1) \, p(w_3|w_1, w_2)...p(w_n|w_1, w_2, ...w_{n-1})$$

School of
**informatics**

# Markov chain

- **Markov assumption**:

  - only previous history matters
  - limited memory: only last $k$ words are included in history
    (older words less relevant)
  $\rightarrow$ $k$**th order Markov model**

- For instance 2-gram language model:

$$p(w_1, w_2, w_3, ..., w_n) = p(w_1)\, p(w_2|w_1)\, p(w_3|w_2)...p(w_n|w_{n-1})$$

- What is conditioned on, here $w_{n-1}$ is called the **history**

School of
**informatics**

# Estimating n-gram probabilities

- We are back in comfortable territory: maximum likelihood estimation

$$p(w_2|w_1) = \frac{count(w_1, w_2)}{count(w_1)}$$

- Collect counts over a large text corpus

- Millions to billions of words are easy to get

School of
**informatics**

# Size of the model

- For each n-gram (e.g. *the big house*), we need to store a probability

- Assuming 20,000 distinct words

| Model | Max. number of parameters |
|---|---|
| 0th order (unigram) | 20,000 |
| 1st order (bigram) | $20,000^2 = 400$ million |
| 2nd order (trigram) | $20,000^3 = 8$ trillion |
| 3rd order (4-gram) | $20,000^4 = 160$ quadrillion |

- In practice, 3-gram LMs are typically used

School of
**informatics**

# Size of model: practical example

- Trained on 10 million sentences from the Gigaword corpus (text collection from New York Times, Wall Street Journal, and news wire sources), about 275 million words.

| 1-gram | 716,706 |
|--------|--------------|
| 2-gram | 12,537,755 |
| 3-gram | 22,174,483 |

- Worst case for number of distinct n-grams is linear with the corpus size.

School of
**informatics**

# How good is the LM?

- A good model assigns a text of real English a high probability

- This can be also measured with cross entropy:

$$H(W) = \frac{1}{n} \log p(W_1^n)$$

- Or, **perplexity**

$$perplexity(W) = 2^{H(W)}$$

# Training set and test set

- We learn the language model from a **training set**, i.e. we collect statistics for n-grams over that sample and estimate the conditional n-gram probabilities.

- We evaluate the language model on a hold-out test set

  - much smaller than training set (thousands of words)
  - not part of the training set!

- We measure perplexity on the test set to gauge the quality of our language model.

# Example: unigram

- Training set

| there is a big house |
| --- |
| i buy a house |
| they buy the new house |

- Model

$$p(there) = 0.0714 \qquad p(is) = 0.0714 \qquad p(a) = 0.1429$$
$$p(big) = 0.0714 \qquad p(house) = 0.2143 \qquad p(i) = 0.0714$$
$$p(buy) = 0.1429 \qquad p(they) = 0.0714 \qquad p(the) = 0.0714$$
$$p(new) = 0.0714$$

- Test sentence $S$: *they buy a big house*

- $p(S) = \underbrace{0.0714}_{they} \times \underbrace{0.1429}_{buy} \times \underbrace{0.0714}_{a} \times \underbrace{0.1429}_{big} \times \underbrace{0.2143}_{house} = 0.0000231$

School of **informatics**

# Example: bigram

- Training set

| |
|---|
| *there is a big house* |
| *i buy a house* |
| *they buy the new house* |

- Model

$$
\begin{array}{lll}
p(big|a) = 0.5 & p(is|there) = 1 & p(buy|they) = 1 \\
p(house|a) = 0.5 & p(buy|i) = 1 & p(a|buy) = 0.5 \\
p(new|the) = 1 & p(house|big) = 1 & p(the|buy) = 0.5 \\
p(a|is) = 1 & p(house|new) = 1 & p(they| <s>) = .333
\end{array}
$$

- Test sentence $S$: *they buy a big house*

- $p(S) = \underbrace{0.333}_{they} \times \underbrace{1}_{buy} \times \underbrace{0.5}_{a} \times \underbrace{0.5}_{big} \times \underbrace{1}_{house} = 0.0833$

# Unseen events

- Another example sentence $S_2$: *they buy a new house.*

- Bigram *a new* has never been seen before

- $p(new|a) = 0 \rightarrow p(S_2) = 0$

- ... but it is a good sentence!

School of **informatics**

# Two types of zeros

- Unknown words

  - handled by an UNKNOWN word token

- Unknown n-grams

  - smoothing by giving them some low probability
  - back-off to lower order n-gram model

- Giving probability mass to unseen events reduces available probability mass for seen events $\Rightarrow$ not maximum likelihood estimates anymore

# Add-one smoothing

For all possible n-grams, add the count of one.
Example:

| bigram | count | $\rightarrow p(w_2|w_1)$ | count+1 | $\rightarrow p(w_2|w_1)$ |
|--------|-------|-------------------------|---------|-------------------------|
| *a big* | 1 | 0.5 | 2 | 0.18 |
| *a house* | 1 | 0.5 | 2 | 0.18 |
| *a new* | 0 | 0 | 1 | 0.09 |
| *a the* | 0 | 0 | 1 | 0.09 |
| *a is* | 0 | 0 | 1 | 0.09 |
| *a there* | 0 | 0 | 1 | 0.09 |
| *a buy* | 0 | 0 | 1 | 0.09 |
| *a a* | 0 | 0 | 1 | 0.09 |
| *a i* | 0 | 0 | 1 | 0.09 |

School of
**informatics**

# Add-one smoothing

- This is Bayesian estimation with a uniform prior.
  Recall: $argmax_M P(M|D) = argmax_M P(D|M) \times P(M)$

- Is too much probability mass wasted on unseen events?
  $\leftrightarrow$ Are impossible/unlikely events estimated too high?

- How can we measure this?

# Expected counts and test set counts

Church and Gale (1991a) experiment: 22 million words training, 22 million words testing, from same domain (AP news wire), counts of bigrams:

| Frequency $r$ in training | Actual frequency in test | Expected frequency in test (add one) |
|:---:|:---:|:---:|
| 0 | 0.000027 | 0.000132 |
| 1 | 0.448 | 0.000274 |
| 2 | 1.25 | 0.000411 |
| 3 | 2.24 | 0.000548 |
| 4 | 3.23 | 0.000685 |
| 5 | 4.21 | 0.000822 |

We overestimate 0-count bigrams ($0.000132 > 0.000027$), but since there are so many, they use up so much probability mass that hardly any is left.

School of
**informatics**

# Using held-out data

- We know from the test data, how much probability mass should be assigned to certain counts.

- We can not use the test data for estimation, because that would be cheating.

- Divide up the training data: one half for count collection, one have for collecting frequencies in unseen text.

- Both halves can be switched and results combined to not lose out on training data.

School of
informatics

# Deleted estimation

- Counts in training $C_t(w_1, ..., w_n)$

- Counts how often an ngram seen in training is seen in held-out training $C_h(w_1, ..., w_n)$

- Number of ngrams with training count $r$: $N_r$

- Total times ngrams of training count $r$ seen in held-out data: $T_r$

- Held-out estimator:

$$p_h(w_1, ..., w_n) = \frac{T_r}{N_r N} \quad \text{where } count(w_1, ..., w_n) = r$$

School of
**informatics**

# Using both halves

- Both halves can be switched and results combined to not lose out on training data

$$p_h(w_1, ..., w_n) = \frac{T_r^{01} + T_r^{10}}{N(N_r^{01} + N_r^{10})} \text{ where } count(w_1, ..., w_n) = r$$