#### Probabilistic Language Models

#### Miles Osborne

School of Informatics University of Edinburgh miles@inf.ed.ac.uk

February 26, 2008



2 A simple language model





# What is a language model?

- A language model is a distribution over sentences:
  - Is the sentence *John loves Mary* more likely than the string *Mary Mary John*?
  - Which word is more likely to be next: John loves ....
- LMs have a wide variety of uses:
  - In Speech Recognition (predict which word is going to be spoken).
  - In Mobile Phones (predict which letter is going to be typed)
  - Machine translation (order good translations from bad ones).
  - In theories of human language learning.

# A simple LM

A first attempt at a LM:

- List all possible sentences we want to consider.
- Assign each sentence a probability.
- Make sure that all probabilities sum to one (etc).

This is no good:

- Languages are usually thought to be infinite.
  - A list of all sentences is going to be big ...
- We hardly ever see all possible sentences.
- What happens when we want to partially process them?

## A simple LM

A second attempt at a LM:

- Assume a sentence consists of words  $w_1 \ldots w_n$ .
- According to the Chain Law:

$$P(w_1, w_2, \ldots, w_n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_2, w_1) \ldots$$

- Some notation: we call the conditioning sequence of words the *history*.
- This suggests that we can decompose a sentence probability as a product of smaller probabilities.

# A simple LM

The key application of the Chain rule is as follows:

- If predicting the current word is independent of some word in the history, then we can drop it.
- Dropping words simplifies a probability.

#### History Simplification

Suppose the history consists of three words  $w_i, w_{i-1}, w_{i-2}$ . Instead of predicting  $w_{i+1}$  using  $P(w_{i+1} | w_i, w_{i-1}, w_{i-2})$ , we can use  $P(w_{i+1} | w_i, w_{i-1})$ 

This is called making a Markov assumption

# A simple LM

Making a Markov assumption simplifies the model:

- Instead of considering all previous words when predicting a word, we only consider a shorter history.
- Estimating word probabilites is a lot easier (as we shall see).
- A Markov assumption may actually have a cognitive basis:
  - Does the first word of a sentence have any impact upon predicting the 40<sup>th</sup> word?
- A Markov assumption technically means assuming conditional independence.

#### A simple LM

A typical Markov assumption uses just the previous two words:

- $P(w_{i+1} | w_i, w_{i-1}).$
- This is called a *trigram* model.
- (When we use just one word, we have a *bigram* model).
- Big language models can use 8-grams.

## A simple LM

Our trigram LM is now:

$$P(w_1, w_2, \ldots, w_n) \approx P(w_1)P(w_2 \mid w_1) \prod_{i=3}^n P(w_i \mid w_{i-1}, w_{i-2})$$

Note: to avoid underflow, we typically take logs instead:

$$\log P(w_1, w_2, \dots, w_n) \approx \log P(w_1) + \log P(w_2 \mid w_1) \sum_{i=3}^n \log P(w_i \mid w_{i-1}, w_{i-2})$$

# A simple LM

Where do the probabilities come from?

- We need to *estimate* probabilities.
- We use a *training* set of sentences.
- A simple approach is to *count*:

#### Estimating coin probabilities

- Suppose we have a coin. To estimate the probability of a head, we throw our coin many times and count the number of heads.
- We also count the number of tails.
- The probability of a head is simply the number of heads divided by the total number of flips.

## A simple LM

We can use the same reasoning for estimating ngram probabilities:

$$P(w_{i+1} \mid w_i, w_{i-1}) = \frac{\mathsf{c}(w_{i+1}, w_i, w_{i-1})}{\mathsf{c}(w_i, w_{i-1})}$$

- Note:  $c(w_i, w_{i-1}) = \sum_j c(w_j, w_i, w_{i-1})$
- Here  $c(\cdot)$  counts the number of times a given ngram occurs.

#### A simple LM

The more often we see a trigram, the more confident we are in the estimated probability:

- If we flipped our coin once, we would estimate the probability of a head as being either 1 or 0 (ouch).
- Flipping a coin more times increases the estimated accuracy.
- The same reasoning applies when estimating trigrams (bigrams etc).

# A simple LM



# Smoothing

N-gram models can be estimated from *trillions* of words!

- Manipulating models this large takes serious engineering.
- ... Using ever more data is good, but it will not always help:
  - What happens when we encounter some novel word (sequence)?
  - What happens when we don't see some word sequence very often?

Smoothing refers to the task of improving estimation.

# Smoothing

If we encounter some word (or word sequence) we never saw, we will assign the entire sentence a probability of zero.

• We may never have seen a laughing cat in our training set:

$$\begin{array}{ll} \mathsf{c}(\mathsf{laughed},\mathsf{cat},\mathsf{the}) = 0 \\ P(\mathsf{laughed} \mid \mathsf{cat},\mathsf{the}) = 0 \\ P(\mathsf{the},\,\mathsf{cat},\,\mathsf{laughed}) = & \dots P(\mathsf{laughed} \mid \mathsf{cat},\mathsf{the}) \\ &= 0 \end{array}$$

This is clearly wrong.

# Smoothing

A simple idea is to pretend we saw all possible trigrams once:

 $\begin{aligned} \mathsf{count}(\mathsf{laughed},\mathsf{cat},\mathsf{the}) &= 1\\ \mathsf{count}(\mathsf{cat},\mathsf{the}) &= \mathit{N} + \mathit{M} \end{aligned}$ 

• Here, *N* was the number of times we actually saw the bigram *the cat.* 

• *M* corresponds to the sum of all the hallucinated single counts. This is called *add-one* smoothing.

# Smoothing

Backoff smoothing: instead of using a trigram model, at times use the corresponding *bigram* model (etc):

$$P(w_{i+1} \mid w_i, w_{i-1})^* = \begin{cases} P(w_{i+1} \mid w_i, w_{i-1}) & \text{if } c(w_{i+1}, w_i, w_{i-1}) > 0 \\ P(w_{i+1} \mid w_i)^* & \text{otherwise} \end{cases}$$

- Intuition: short ngrams will be seen more often than longer ones.
- Shorter ngrams are more reliably estimated than longer ones.
- (But, simpler ngrams can be less accurate)



- Probabilistic language models are distributions over sentences.
- The Chain Rule allows for practical modelling.
- Parameters can be estimated by counting.
- Smoothing is vital in practice.