The Noisy Channel Model and Markov Models

Mark Johnson

September 3, 2014

MACQUARIE UNIVERSITY

The big ideas

- The story so far:
  - machine learning classifiers learn a function that maps a data item X to a label Y
  - handle *large item spaces* $\mathcal{X}$ by decomposing each data item X into a *collection of features* $\boldsymbol{F}$
  - but the *label spaces* $\mathcal{Y}$ have to be small to avoid sparse data problems
- Where we're going from here:
  - many important problems involve *large label spaces* $\mathcal{Y}$
    - Part-Of-Speech (POS) tagging, where Y is a *vector of POS tags*
    - Syntactic parsing, where Y is a *syntactic parse tree*
  - basic approach: decompose Y into parts or features $\boldsymbol{G}$
  - if each feature G_j is *independent*, just learn a separate classifier for each G_j
  - but often there are *important dependencies* between the G_j
    - in POS tagging, adjectives typically precede nouns

$\Rightarrow$ more sophisticated models are required to capture these dependencies

Outline

The noisy channel model

Bigram language models

Learning bigram models from text

Markov Chains and higher-order *n*-grams

n-gram language models as Bayes nets

Summary

Machine learning in computational linguistics

- Many problems involve predicting a complex label Y from data X
  - in *automatic speech recognition*, X is acoustic waveform, Y is transcript
  - in *machine translation*, X is source language text, Y is target language translation
  - in *spelling correction*, X is a source text with spelling mistakes, and Y is a target text without spelling mistakes
  - in *automatic summarisation*, X is a document, Y is a summary of that document
- Suppose we can estimate $\mathrm{P}(Y \mid X)$ somehow. Then we should compute:

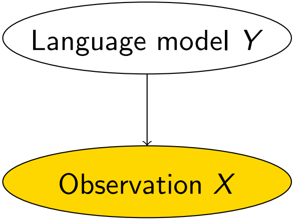
$$\widehat{y}(x) \quad = \quad \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \mathrm{P}(Y{=}y \mid X{=}x)$$

- Problems we have to solve:
  - $|\mathcal{Y}|$ is astronomical $\Rightarrow$ computing argmax may be intractable
  - $|\mathcal{Y}|$ is astronomical $\Rightarrow$ *how can we estimate $\mathrm{P}(Y \mid X)$?*

The noisy channel model

- The noisy channel model uses *Bayes rule* to invert $\mathrm{P}(Y \mid X)$

$$\mathrm{P}(Y \mid X) \quad = \quad \frac{\mathrm{P}(X \mid Y)\,\mathrm{P}(Y)}{\mathrm{P}(X)}$$



- We can ignore $\mathrm{P}(X)$ if our goal is to compute

$$\widehat{y}(x) \quad = \quad \operatorname*{argmax}_{y \in \mathcal{Y}} \mathrm{P}(X{=}x \mid Y{=}y)\,\mathrm{P}(Y{=}y)$$

  because $\mathrm{P}(X{=}x)$ is a constant

- $\mathrm{P}(X \mid Y)$ is called the *channel model* or the *distortion model*
- $\mathrm{P}(Y)$ is called the *source model* or the *language model*
  - can often be learnt from cheap, readily available data

The noisy channel model in spelling correction, speech recognition and machine translation

$$\widehat{y}(x) \quad = \quad \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \; \underbrace{\mathrm{P}(X{=}x \mid Y{=}y)}_{\text{channel model}} \; \underbrace{\mathrm{P}(Y{=}y)}_{\text{source model}}$$

- The *channel models* are task-specific
  - for spelling correction, $\mathrm{P}(X \mid Y)$ maps words to their likely mis-spellings
  - for speech recognition, $\mathrm{P}(X \mid Y)$ maps words or phonemes to acoustic waveforms
  - for machine translation, $\mathrm{P}(X \mid Y)$ maps words or phrases to their translations
- The *source model* $\mathrm{P}(Y)$, which is also called a *language model*, is the same
  - $\mathrm{P}(Y)$ is the *probability of a sentence Y*
  - can be learned from readily available text corpora

MACQUARIE UNIVERSITY

Outline

The noisy channel model

Bigram language models

Learning bigram models from text

Markov Chains and higher-order *n*-grams

n-gram language models as Bayes nets

Summary

Language models

- A *language model* calculates the probability of a sequence of words or phonemes
- In many NLP applications the *source model* $\mathrm{P}(Y)$ in a noisy channel model is a language model

$$\widehat{y}(x) \quad = \quad \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \; \underbrace{\mathrm{P}(X{=}x \mid Y{=}y)}_{\text{channel model}} \; \underbrace{\mathrm{P}(Y{=}y)}_{\text{source model}}$$

- In such applications, Y is a *sequence of words or phonemes*
- The language model is used to calculate the probability of possible sequences Y of words or phonemes
- The job of the language model is to distinguish likely sequences of words or phonemes from unlikely ones
- It can be learnt from cheaply-available text collections

The bigram assumption for language models

- A *language model* estimates the probability $\mathrm{P}(\boldsymbol{Y})$ of a sentence $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$, where Y_i is the ith word in the sentence
- Recall the relationship between joint and conditional probabilities:

$$\mathrm{P}(U, V) = \mathrm{P}(V)\,\mathrm{P}(U \mid V)$$

- We can use this to rewrite $\mathrm{P}(\boldsymbol{Y}) = \mathrm{P}(Y_1, \ldots, Y_n)$:

$$\begin{aligned}\mathrm{P}(Y_1, \ldots, Y_n) &= \mathrm{P}(Y_1)\,\mathrm{P}(Y_2, \ldots, Y_n \mid Y_1) \\ &= \mathrm{P}(Y_1)\,\mathrm{P}(Y_2 \mid Y_1)\,\mathrm{P}(Y_3 \mid Y_1, Y_2) \\ &\quad \ldots\,\mathrm{P}(Y_n \mid Y_1, \ldots, Y_{n-1})\end{aligned}$$

- Now make the *bigram assumption*: $\mathrm{P}(Y_j \mid Y_1, \ldots, Y_{j-1}) \approx \mathrm{P}(Y_j \mid Y_{j-1})$, i.e., word Y_j only depends on Y_{j-1}
- Then $\mathrm{P}(\boldsymbol{Y})$ simplifies to:

$$\mathrm{P}(Y_1, \ldots, Y_n) = \mathrm{P}(Y_1)\,\mathrm{P}(Y_2 \mid Y_1)\,\ldots\,\mathrm{P}(Y_n \mid Y_{n-1})$$

MACQUARIE UNIVERSITY

Homogeneity assumption in language models

- Using bigram assumption we simplified

$$\begin{aligned} \mathrm{P}(Y_1, \ldots, Y_n) &= \mathrm{P}(Y_1)\, \mathrm{P}(Y_2 \mid Y_1) \ldots \mathrm{P}(Y_n \mid Y_{n-1}) \\ &= \mathrm{P}(Y_1) \prod_{i=2}^{n} \mathrm{P}(Y_i \mid Y_{i-1}) \end{aligned}$$

- *Homogeneity assumption*: conditional probabilities *don't change with i*, i.e., there is a matrix $\boldsymbol{s}$ such that

$$\mathrm{P}(Y_i = y \mid Y_{i-1} = y') = s_{y,y'}$$

- Then the *bigram model probability* $\mathrm{P}(\boldsymbol{Y}{=}\boldsymbol{y})$ *of a sentence* $\boldsymbol{y} = (y_1, \ldots, y_n)$ is:

$$\begin{aligned} \mathrm{P}(\boldsymbol{Y}{=}\boldsymbol{y}) &= \mathrm{P}(Y_1 = y_1)\, s_{y_2,y_1}\, s_{y_3,y_2} \ \ldots \ s_{y_n,y_{n-1}} \\ &= \mathrm{P}(Y_1 = y_1) \prod_{i=2}^{n} s_{y_i,y_{i-1}} \end{aligned}$$

Using end-markers to handle initial and final conditions

- We simplify the model *by assuming the string $\boldsymbol{y}$ is padded with end-markers* \$

$$\boldsymbol{y} = (\$, y_1, y_2, \ldots, y_n, \$)$$

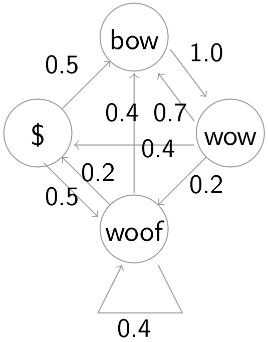
I.e., $Y_0 = \$$ and $Y_{n+1} = \$$

- Then the bigram language model probability $\mathrm{P}(\boldsymbol{Y} = \boldsymbol{y})$ is:

$$\begin{aligned}\mathrm{P}(\boldsymbol{Y} = \boldsymbol{y}) &= s_{y_1,\$}\, s_{y_2,y_1} \ldots s_{y_n,y_{n-1}}\, s_{\$,y_n} \\ &= \prod_{i=1}^{n+1} s_{y_i,y_{i-1}}\end{aligned}$$

Bigram language model example

$$\boldsymbol{s} = \begin{array}{c|cccc} y_i \backslash y_{i-1} & \$ & bow & wow & woof \\ \hline \$ & 0 & 0 & 0.1 & 0.2 \\ bow & 0.5 & 0 & 0.7 & 0.4 \\ wow & 0 & 1.0 & 0 & 0 \\ woof & 0.5 & 0 & 0.2 & 0.4 \end{array}$$



$$\begin{aligned} &\mathrm{P}(\$,\mathrm{bow},\mathrm{wow},\mathrm{woof},\mathrm{woof},\$) \\ &= s_{\mathrm{bow},\$}\ s_{\mathrm{wow},\mathrm{bow}}\ s_{\mathrm{woof},\mathrm{wow}}\ s_{\mathrm{woof},\mathrm{woof}}\ s_{\$,\mathrm{woof}} \\ &= 0.5 \cdot 1.0 \cdot 0.2 \cdot 0.4 \cdot 0.2 \\ &= 0.008 \end{aligned}$$

Outline

The noisy channel model

Bigram language models

Learning bigram models from text

Markov Chains and higher-order *n*-grams

n-gram language models as Bayes nets

Summary

Estimating bigram models from text

- We can estimate the bigram model $\boldsymbol{s}$ from a *text corpus*
- Collect a *vector of unigram counts* $\boldsymbol{m}$ and a *matrix of bigram counts* $\boldsymbol{n}$

$$\begin{aligned} m_y &= \text{number of times } y \text{ is followed by anything in corpus} \\ n_{y',y} &= \text{number of times } y' \text{ follows } y \text{ in corpus} \end{aligned}$$

  - make sure you count the beginning and end of sentence markers!
- Maximum likelihood estimates:

$$\hat{s}_{y',y} = \frac{n_{y',y}}{m_y}$$

- Add-1 smoothed estimates (a good idea!):

$$\hat{\hat{s}}_{y',y} = \frac{n_{y',y} + 1}{m_y + |\mathcal{V}|}$$

where $\mathcal{V}$ is the *vocabulary* (set of words) of the corpus

MACQUARIE UNIVERSITY

Estimating a bigram model example

$$\text{corpus} = \begin{pmatrix} \text{\$ bow wow woof woof \$} \\ \text{\$ woof bow wow bow wow woof \$} \\ \text{\$ bow wow \$} \end{pmatrix}$$

$$\mathcal{V} = \{\$, \text{bow}, \text{wow}, \text{woof}\}$$

$$\boldsymbol{m} = \begin{array}{cccc} \$ & \textit{bow} & \textit{wow} & \textit{woof} \\ 3 & 4 & 4 & 4 \end{array}$$

$$\boldsymbol{n} = \begin{array}{c|cccc} y_i \backslash y_{i-1} & \$ & \textit{bow} & \textit{wow} & \textit{woof} \\ \hline \$ & 0 & 0 & 1 & 2 \\ \textit{bow} & 2 & 0 & 1 & 1 \\ \textit{wow} & 0 & 4 & 0 & 0 \\ \textit{woof} & 1 & 0 & 2 & 1 \end{array}$$

$$\hat{\boldsymbol{s}} = \begin{array}{c|cccc} y_i \backslash y_{i-1} & \$ & \textit{bow} & \textit{wow} & \textit{woof} \\ \hline \$ & 1/7 & 1/8 & 2/8 & 3/8 \\ \textit{bow} & 3/7 & 1/8 & 2/8 & 2/8 \\ \textit{wow} & 1/7 & 5/8 & 1/8 & 1/8 \\ \textit{woof} & 2/7 & 1/8 & 3/8 & 2/8 \end{array}$$

Outline

The noisy channel model

Bigram language models

Learning bigram models from text

Markov Chains and higher-order *n*-grams

n-gram language models as Bayes nets

Summary

MACQUARIE UNIVERSITY

Markov Models

- A *first-order Markov chain* is a sequence of random variables $Y_1, Y_2, Y_3, \ldots$, where:

$$\mathrm{P}(Y_i \mid Y_1, \ldots, Y_{i-1}) \;=\; \mathrm{P}(Y_i \mid Y_{i-1})$$

  I.e., Y_i is independent of $Y_1, \ldots, Y_{i-2}$ given Y_i, or
  the value Y_i at time i only depends on the value Y_{i-1} at time $i-1$
- The bigram language model is a first-order Markov chain
  - Informally, the order of a Markov chain indicates "how far back in the past" the next state can depend on

MACQUARIE UNIVERSITY

Higher-order Markov chains and *n*-gram language models

- An *n*-gram language model uses adjacent *n*-word sequences to predict the probability of a sequence
  - E.g., the *trigrams* in $\boldsymbol{y} = (\mathrm{the}, \mathrm{rain}, \mathrm{in}, \mathrm{spain})$ are $(\$, \$, \mathrm{the})$, $(\$, \mathrm{the}, \mathrm{rain})$, $(\mathrm{the}, \mathrm{rain}, \mathrm{in})$, $(\mathrm{rain}, \mathrm{in}, \mathrm{spain})$, $(\mathrm{in}, \mathrm{spain}, \$)$, $(\mathrm{spain}, \$, \$)$
- In an *m*th order Markov chain, Y_i depends only on $Y_{i-m}, \ldots, Y_{i-1}$
- So an $m + 1$-gram language model is an *m*th order Markov chain
  - E.g., a bigram language model is a first-order Markov chain
  - E.g., a trigram language model is a second-order Markov chain

MACQUARIE UNIVERSITY

Outline

The noisy channel model

Bigram language models

Learning bigram models from text

Markov Chains and higher-order *n*-grams

n-gram language models as Bayes nets

Summary

n-gram language models

- Goal: estimate $\mathrm{P}(\boldsymbol{y})$, where $\boldsymbol{y} = (y_1, \ldots, y_m)$ is a sequence of words
- *n*-gram models decompose $\mathrm{P}(\boldsymbol{y})$ into product of conditional distributions

$$\mathrm{P}(\boldsymbol{y}) = \mathrm{P}(y_1)\mathrm{P}(y_2 \mid y_1)\mathrm{P}(y_3 \mid y_1, y_2) \ldots \mathrm{P}(y_m \mid y_1, \ldots, y_{m-1})$$

E.g.,
$$\begin{aligned}\mathrm{P}(\textit{wreck a nice beach}) &= \mathrm{P}(\textit{wreck})\,\mathrm{P}(\textit{a} \mid \textit{wreck})\,\mathrm{P}(\textit{nice} \mid \textit{wreck a}) \\ &\quad \mathrm{P}(\textit{beach} \mid \textit{wreck a nice})\end{aligned}$$

- *n*-gram assumption: *no dependencies span more than n words*, i.e.,

$$\mathrm{P}(y_i \mid y_1, \ldots, y_{i-1}) \approx \mathrm{P}(y_i \mid y_{i-n}, \ldots, y_{i-1})$$

E.g., A *bigram model* is an *n*-gram model where $n = 2$:

$$\begin{aligned}\mathrm{P}(\textit{wreck a nice beach}) &\approx \mathrm{P}(\textit{wreck})\,\mathrm{P}(\textit{a} \mid \textit{wreck})\,\mathrm{P}(\textit{nice} \mid \textit{a}) \\ &\quad \mathrm{P}(\textit{beach} \mid \textit{nice})\end{aligned}$$

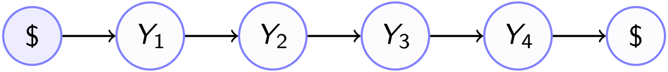
MACQUARIE UNIVERSITY

n-gram language models as Markov models and Bayes nets

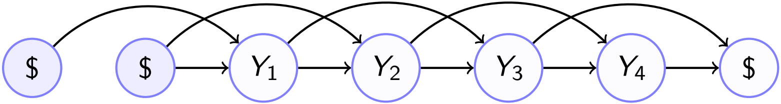
- An *n*-gram language model is a *Markov model* that *factorises the distribution over sentences into a product of conditional distributions*:

$$\mathrm{P}(\boldsymbol{y}) \quad = \quad \prod_{i=1}^{m} \mathrm{P}(y_i \mid y_{i-n}, \ldots, y_{i-1})$$

  - *pad $\boldsymbol{y}$ with end markers*, i.e., $\boldsymbol{y} = (\$, y_1, y_2, \ldots, y_m, \$)$
- Bigram language model as Bayes net:



- Trigram language model as Bayes net:



MACQUARIE UNIVERSITY

The conditional word models in *n*-gram models

- An *n*-gram model factorises $\mathrm{P}(\boldsymbol{y})$ into a product of conditional models, each of the form:

$$\mathrm{P}(y_n \mid y_1, \ldots, y_{n-1})$$

- The performance of an *n*-gram model depends greatly on exactly how these conditional models are defined
  - huge amount of work on this
- *Random forest* and *deep learning* methods for estimating these conditional distributions currently produce state-of-the-art language models

Outline

The noisy channel model

Bigram language models

Learning bigram models from text

Markov Chains and higher-order *n*-grams

n-gram language models as Bayes nets

Summary

MACQUARIE UNIVERSITY

Summary

- In many NLP applications, the label space $\mathcal{Y}$ is *astronomically large*
  $\Rightarrow$ decompose it into components (e.g., elements, features, etc.)
- The *noisy channel model* uses Bayes rule to "invert" a sequence labelling problem
  - enables us to use *language models* trained on large text collections
- n-gram language models are *Markov models* that predict the next word based on the preceding $n-1$ words