Introduction to Natural Language Processing I
[Statistické metody zpracování přirozených jazyků I]
(NPFL067)
http://ufal.mff.cuni.cz/courses/npfl067

prof. RNDr. Jan Hajič, Dr. / doc. RNDr. Pavel Pecina, Ph.D.
ÚFAL MFF UK
{hajic,pecina}@ufal.mff.cuni.cz
http://ufal.mff.cuni.cz/jan-hajic
Language Modeling
(and the Noisy Channel)
The Noisy Channel

• Prototypical case:

\[
\begin{array}{c}
\text{Input} \\
0,1,1,1,0,1,0,1,\ldots
\end{array} \rightarrow \begin{array}{c}
\text{The channel} \\
(\text{adds noise})
\end{array} \rightarrow \begin{array}{c}
\text{Output (noisy)} \\
0,1,1,0,0,1,1,0,\ldots
\end{array}
\]

• Model: probability of error (noise):

\[p(0|1) = .3 \quad p(1|1) = .7 \quad p(1|0) = .4 \quad p(0|0) = .6\]

• The Task:

known: the noisy output; want to know: the input (decoding)
Noisy Channel Applications

• **OCR**
  - straightforward: text $\rightarrow$ print (adds noise), scan $\rightarrow$ image

• **Handwriting recognition**
  - text $\rightarrow$ neurons, muscles ("noise"), scan/digitize $\rightarrow$ image

• **Speech recognition (dictation, commands, etc.)**
  - text $\rightarrow$ conversion to acoustic signal ("noise") $\rightarrow$ acoustic waves

• **Machine Translation**
  - text in target language $\rightarrow$ translation ("noise") $\rightarrow$ source language

• **Also: Part of Speech Tagging**
  - sequence of tags $\rightarrow$ selection of word forms $\rightarrow$ text
Noisy Channel: The Golden Rule of ...

• Recall:

\[ p(A|B) = \frac{p(B|A) \ p(A)}{p(B)} \]  (Bayes formula)

\[ A_{\text{best}} = \text{argmax}_A p(B|A) \ p(A) \]  (The Golden Rule)

• \( p(B|A) \): the acoustic/image/translation/lexical model
  – application-specific name
  – will explore later

• \( p(A) \): \textit{the language model}
The Perfect Language Model

- **Sequence of word forms** [forget about tagging for the moment]
- **Notation:** \( A \sim W = (w_1, w_2, w_3, \ldots, w_d) \)
- **The big (modeling) question:**
  \[ p(W) = ? \]
- **Well, we know (Bayes/chain rule →):**
  \[ p(W) = p(w_1, w_2, w_3, \ldots, w_d) = p(w_1) \times p(w_2|w_1) \times p(w_3|w_1, w_2) \times \ldots \times p(w_d|w_1, w_2, \ldots, w_{d-1}) \]
- **Not practical (even short \( W \) → too many parameters)**
Markov Chain

- Unlimited memory (cf. previous foil):
  - for $w_i$, we know all its predecessors $w_1, w_2, w_3, ..., w_{i-1}$
- Limited memory:
  - we disregard “too old” predecessors
  - remember only $k$ previous words: $w_{i-k}, w_{i-k+1}, ..., w_{i-1}$
  - called “$k^{th}$ order Markov approximation”
- + stationary character (no change over time):
  $$p(W) \equiv \prod_{i=1}^{d} p(w_i|w_{i-k}, w_{i-k+1}, ..., w_{i-1}), \quad d = |W|$$
n-gram Language Models

• \((n-1)^{th}\) order Markov approximation \(\rightarrow\) n-gram LM:

\[
p(W) = \frac{df}{\prod_{i=1..d} p(w_i|w_{i-n+1},w_{i-n+2},...,w_{i-1})}
\]

• In particular (assume vocabulary \(|V| = 60k\)):
  - 0-gram LM: uniform model, \(p(w) = 1/|V|\), 1 parameter
  - 1-gram LM: unigram model, \(p(w)\), \(6\times10^4\) parameters
  - 2-gram LM: bigram model, \(p(w_i|w_{i-1})\) \(3.6\times10^9\) parameters
  - 3-gram LM: trigram model, \(p(w_i|w_{i-2},w_{i-1})\) \(2.16\times10^{14}\) parameters
LM: Observations

• How large $n$?
  – nothing is enough (theoretically)
  – but anyway: as much as possible (→ close to “perfect” model)
  – empirically: 3
    • parameter estimation? (reliability, data availability, storage space, ...)
    • 4 is too much: $|V| = 60k \rightarrow 1.296 \times 10^{19}$ parameters
    • but: 6-7 would be (almost) ideal (having enough data): in fact, one can recover the original text sequence from 7-grams!

• Reliability ~ (1 / Detail) (→ need compromise)
• For now, keep word forms (no “linguistic” processing)
The Length Issue

• $\forall n; \sum_{w \in \Omega^n} p(w) = 1 \Rightarrow \sum_{n=1..\infty} \sum_{w \in \Omega^n} p(w) >> 1 (\rightarrow \infty)$

• We want to model all sequences of words
  – for “fixed” length tasks: no problem - n fixed, sum is 1
    • tagging, OCR/handwriting (if words identified ahead of time)
  – for “variable” length tasks: have to account for
    • discount shorter sentences

• General model: for each sequence of words of length n,
  define $p'(w) = \lambda_n p(w)$ such that $\sum_{n=1..\infty} \lambda_n = 1 \Rightarrow \sum_{n=1..\infty} \sum_{w \in \Omega^n} p'(w) = 1$
  e.g., estimate $\lambda_n$ from data; or use normal or other distribution
Parameter Estimation

- Parameter: numerical value needed to compute \( p(w|h) \)
- From data (how else?)
- Data preparation:
  - get rid of formatting etc. ("text cleaning")
  - define words (separate but include punctuation, call it "word")
  - define sentence boundaries (insert "words" \(<s>\) and \(</s>\))
  - letter case: keep, discard, or be smart:
    - name recognition
    - number type identification
    [these are huge problems per se!]
  - numbers: keep, replace by \(<\text{num}>\), or be smart (form ~ pronunciation)
Maximum Likelihood Estimate

• MLE: Relative Frequency...
  – ...best predicts the data at hand (the “training data”)

• Trigrams from Training Data T:
  – count sequences of three words in T: $c_3(w_{i-2}, w_{i-1}, w_i)$
    [NB: notation: just saying that the three words follow each other]
  – count sequences of two words in T: $c_2(w_{i-1}, w_i)$:
    • either use $c_2(y,z) = \sum_w c_3(y,z,w)$
    • or count differently at the beginning (& end) of data!

\[
p(w_i|w_{i-2}, w_{i-1}) = \text{est. } \frac{c_3(w_{i-2}, w_{i-1}, w_i)}{c_2(w_{i-2}, w_{i-1})} \]

UFAL MFF UK NPFL067/Intro to Statistical NLP I/Jan Hajic - Pavel Pecina 73
Character Language Model

• Use individual characters instead of words:

\[ p(W) = \frac{\prod_{i=1}^{d} p(c_i | c_{i-n+1}, c_{i-n+2}, \ldots, c_{i-1})}{df} \]

• Same formulas etc.
• Might consider 4-grams, 5-grams or even more
• Good only for language comparison
• Transform cross-entropy between letter- and word-based models:

\[ H_S(p_c) = \frac{H_S(p_w)}{\text{avg. \# of characters/word in } S} \]
LM: an Example

• Training data:

\(<s> <s> \text{He can buy the can of soda.}\)

– Unigram: \(p_1(\text{He}) = p_1(\text{buy}) = p_1(\text{the}) = p_1(\text{of}) = p_1(\text{soda}) = p_1(.) = 0.125\)

\(p_1(\text{can}) = 0.25\)

– Bigram: \(p_2(\text{He}|<s>) = 1, p_2(\text{can}|\text{He}) = 1, p_2(\text{buy}|\text{can}) = 0.5,\)

\(p_2(\text{of}|\text{can}) = 0.5, p_2(\text{the}|\text{buy}) = 1,\ldots\)

– Trigram: \(p_3(\text{He}|<s>,<s>) = 1, p_3(\text{can}|<s>,\text{He}) = 1,\)

\(p_3(\text{buy}|\text{He},\text{can}) = 1, p_3(\text{of}|\text{the},\text{can}) = 1,\ldots, p_3(.)|\text{of,soda}) = 1.\)

– Entropy: \(H(p_1) = 2.75, H(p_2) = 0.25, H(p_3) = 0\) ← Great?!
LM: an Example (The Problem)

- Cross-entropy:
- \( S = <s> <s> \) It was the greatest buy of all.
- Even \( H_S(p_1) \) fails (= \( H_S(p_2) = H_S(p_3) = \infty \)), because:
  - all unigrams but \( p_1(\text{the}) \), \( p_1(\text{buy}) \), \( p_1(\text{of}) \) and \( p_1(\text{.}) \) are 0.
  - all bigram probabilities are 0.
  - all trigram probabilities are 0.
- We want: to make all (theoretically possible*) probabilities non-zero.

*in fact, all: remember our graph from day 1?