NPFL147

Statistical Methods in Natural Language Processing

3. Statistical language modeling

Pavel Pecina, Jindřich Helcl

21 October, 2025

Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

CC BY SA
unless otherwise stated

Course Segments

1. Introduction, probability, essential information theory
2. Statistical language modelling (n-gram)
3. Statistical properties of words
4. Word embeddings
5. Hidden Markov models, Tagging

Recap from Last Week

Entropy

- Let $p_X(x)$ be a distribution of random variable X
- Basic outcomes (alphabet) Ω

$$\boxed{H(X) = - \sum_{x \in \Omega} p(x) \log_2 p(x)}$$

- Unit: bits ($\log_e$: nats)
- Notation: $H(X) = H_p(X) = H(p) = H_X(p) = H(p_X)$

Joint Entropy and Conditional Entropy

- Two random variables: X (space Ω), Y (Ψ)
- Joint entropy:
  - no big deal: (X,Y) considered a single event:

$$H(X,Y) = - \sum_{x \in \Omega} \sum_{y \in \Psi} p(x,y) \log_2 p(x,y)$$

- Conditional entropy:

$$H(Y|X) = - \sum_{x \in \Omega} \sum_{y \in \Psi} \underline{p(x,y)} \log_2 p(y|x)$$

  - recall that $H(X) = E(\log_2(1/p_X(x)))$
  - (weighted "average", and weights are not conditional)

Kullback-Leibler Distance (Relative Entropy)

- Remember:
  - long series of experiments … c_i/T_i oscillates around some number… we can only estimate it … to get a distribution q.
- So we get a distribution q; (sample space Ω, r.v. X)
- The true distribution is, however, p (same Ω, X)

⇒ how big error are we making?

- $D(p||q)$ (the Kullback-Leibler distance):

$$D(p||q) = \sum_{x \in \Omega} \underline{p(x)} \log_2 (p(x)/q(x)) = E_p \log_2 (p(x)/q(x))$$

Mutual Information

- Rewrite the definition:
  - recall: $D(r\|s) = \sum_{v \in \Omega} r(v) \log_2 (r(v)/s(v))$;
  - substitute: $r(v) = p(x,y)$, $s(v) = p(x)p(y)$; $<v> \sim <x,y>$

$$
\begin{aligned}
I(X,Y) &= D(p(x,y) \,\|\, p(x)p(y)) = \\
&= \sum_{x \in \Omega} \sum_{y \in \Psi} p(x,y) \log_2 (p(x,y)/p(x)p(y))
\end{aligned}
$$

- Measured in bits (what else? :-)

Information Inequality

$$\boxed{D(p||q) \geq 0}$$

Proof:

$$0 = -\log 1 = -\log \sum_{x\in\Omega} q(x) = -\log \sum_{x\in\Omega} p(x)(q(x)/p(x) \leq$$

...apply Jensen's inequality here ($\underline{-\log}$ is convex)...

$$\leq \sum_{x\in\Omega} p(x)(-\log(q(x)/p(x))) = \sum_{x\in\Omega} p(x)\log(p(x)/q(x)) = D(p||q)$$

Cross Entropy: The Formula

- $H_{p'}(\tilde{p}) = H(p') + D(p' \| \tilde{p})$

$$\boxed{H_{p'}(\tilde{p}) = - \sum_{x \in \Omega} p'(x) \log_2 \tilde{p}(x)}$$

- p' is certainly not the true p, but we can consider it the "real world" distribution against which we test
- note on notation (confusing): $p/p' \leftrightarrow \tilde{p}$, also $H_{T'}(p)$
- (Cross) Perplexity:

$$G_{p'}(\tilde{p}) = G_{T'}(\tilde{p}) = 2^{H_{p'}(\tilde{p})}$$

Sample Space vs. Data

- In practice, it is often inconvenient to sum over the sample space(s) Ψ, Ω (especially for cross entropy!)
- Use the following formula:

$$\boxed{\begin{aligned} H_{p'}(p) = \quad & - \sum_{y \in \Psi,\, x \in \Omega} p'(y,x) \log_2 p(y|x) = \\ & - 1/|T'| \sum_{i = 1 \ldots |T'|} \log_2 p(y_i|x_i) \end{aligned}}$$

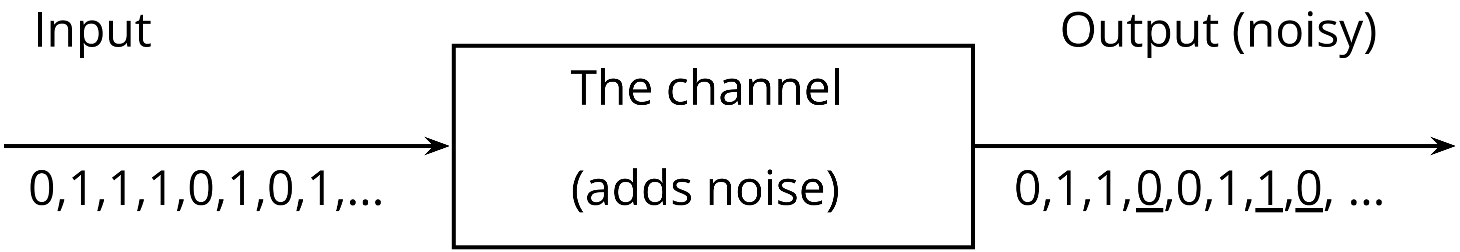
- This is the normalized log probability of the “test” data:

$$H_{p'}(p) = - 1/|T'| \log_2 \prod_{i = 1 \ldots |T'|} p(y_i|x_i)$$

Noisy Channel Model

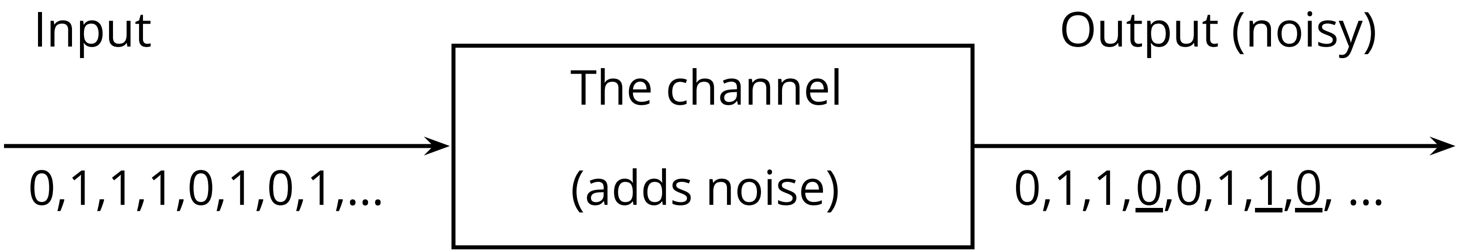
Noisy Channel

Prototypical case:



Noisy Channel

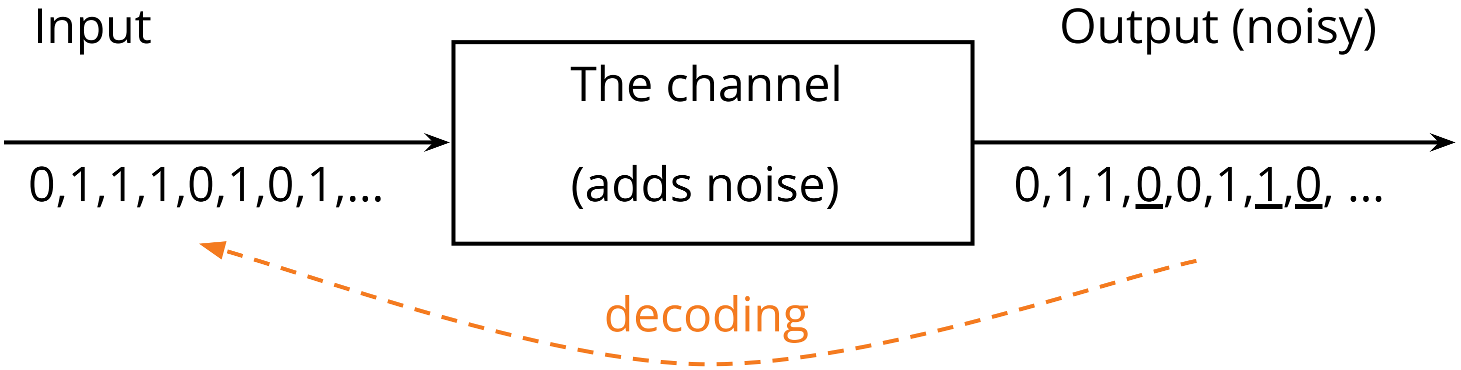
Prototypical case:



- Model: probability of error (noise)
- Example: $p(0|1) = 0.25, \quad p(1|1) = 0.75, \quad p(1|0) = 0.5, \quad p(0|0) = 0.5$
- The Task:
    - known: the noisy output; want to know: the input (decoding)

Noisy Channel

Prototypical case:



- Model: probability of error (noise)
- Example: $p(0|1) = 0.25, \quad p(1|1) = 0.75, \quad p(1|0) = 0.5, \quad p(0|0) = 0.5$
- <u>The Task:</u>
  - known: the noisy output; want to know: the input (<u>decoding</u>)

Noisy Channel Applications

- Optical Character Recognition (OCR)
  - text → *print (adds noise), scan* → image
- Handwriting recognition (HR)
  - text → *neurons, muscles (“noise”), scan/digitize* → image
- Speech recognition (ASR)
  - text → *conversion to acoustic signal (“noise”)* → acoustic waves
- Machine Translation (MT)
  - text in target language → *translation (“noise”)* → text in source language
- Also: Part of Speech Tagging
  - sequence of tags → *selection of word forms* → text

Noisy Channel: The Golden Rule of ... OCR, HR, ASR, MT, ...

- Recall:

$$A_{best} = \text{argmax}_A p(A|B)$$

$$A_{best} = \text{argmax}_A p(B|A)\ \ p(A) / \cancel{p(B)} \qquad \textit{(Bayes Formula)}$$

$$\boxed{A_{best} = \text{argmax}_A\ p(B|A)\ p(A)} \qquad \textit{(The Golden Rule)}$$

- Where:
  - $p(B|A)$: the acoustic/image/translation/lexical model
    - application-specific name
    - will explore later
  - $p(A)$: ***<u>the language model</u>***

The Perfect Language Model

- Sequence of word forms
- 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 $\rightarrow$):

$$p(W) = p(w_1,w_2,w_3,\ldots,w_d) =$$

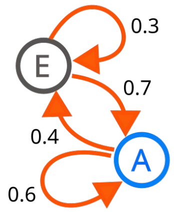
$$= p(w_1) \times p(w_2|w_1) \times p(w_3|w_1,w_2) \times\ldots\times p(w_d|w_1,w_2,\ldots,w_{d-1})$$

- Not practical (even short $W \rightarrow$ too many parameters)

Markov Chain

- Unlimited memory (cf. previous slide):
  - for w_i, we know all its predecessors $w_1, w_2, w_3, \ldots, w_{i-1}$
- Limited memory:
  - we disregard "too old" predecessors
  - remember only k previous words: $w_{i-k}, w_{i-k+1}, \ldots, w_{i-1}$
  - called "kth order Markov approximation"
- + stationary character (no change over time) → Markov Chain:

$$p(W) \cong \Pi_{i=1..d} p(w_i | w_{i-k}, w_{i-k+1}, \ldots, w_{i-1}),\ d = |W|$$



Picture from Wikipedia

n-gram Language Models

n-gram Language Model

- $(n\text{-}1)^{th}$ order Markov approximation → *n-gram Language Model*:

$$p(W) =^{df} \prod_{i=1..d} p(\overbrace{w_i}^{\text{prediction}} | \overbrace{w_{i-n+1}, w_{i-n+2}, \ldots, w_{i-1}}^{\text{history}})$$

n-gram Language Model

- $(n\text{-}1)^{th}$ order Markov approximation → *n-gram Language Model*:

$$p(W) =^{df} \prod_{i=1..d} p(\overset{\text{prediction}}{w_i} | \overbrace{w_{i-n+1}, w_{i-n+2}, \ldots, w_{i-1}}^{\text{history}})$$

- In particular (assume vocabulary $|V| = 60k$):
  - 0-gram LM: uniform model, $p(w) = 1/|V|$, 1 parameter
  - 1-gram LM: unigram model, $p(w)$, 6×10^4 parameters
  - 2-gram LM: bigram model, $p(w_i|w_{i-1})$, 3.6×10^9 parameters
  - 3-gram LM: trigram model, $p(w_i|w_{i-2},w_{i-1})$, 2.16×10^{14} parameters

LM: Observations

- How large n?
  - Nothing is enough (theoretically)
  - But anyway: as much as possible ($\rightarrow$ close to “perfect” model)
  - Neural models allow context/history of thousands of words
  - n-gram models: 3–7
    - 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 ~ $(1 / \text{Detail})$ ($\rightarrow$ need compromise)

The Length Issue

- $\forall n: \Sigma_{w \in \Omega^n} p(w) = 1 \Rightarrow \Sigma_{n=1..\infty} \Sigma_{w \in \Omega^n} p(w) >> 1 \ (\rightarrow \infty)$
- 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*

The Length Issue

- $\forall n: \Sigma_{w \in \Omega^n} p(w) = 1 \Rightarrow \Sigma_{n=1..\infty} \Sigma_{w \in \Omega^n} p(w) >> 1 \ (\rightarrow \infty)$
- 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 $\Sigma_{n=1..\infty} \lambda_n = 1 \Rightarrow \Sigma_{n=1..\infty} \Sigma_{w \in \Omega^n} p'(w) = 1$

  - e.g., estimate λ_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 <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: $c_3(w_{i-2},w_{i-1},w_i)$
    (*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 $\mathbf{c_2(y,z) = \Sigma_w\ c_3(y,z,w)}$
    - or count differently at the beginning (& end) of data!

$$\boxed{p(w_i|w_{i-2},w_{i-1}) =^{est} c_3(w_{i-2},w_{i-1},w_i) \,/\, c_2(w_{i-2},w_{i-1})}$$

Character Language Model

- Use individual characters instead of words:

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

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

$$H_S(p_{char}) = H_S(p_{word}) \;/\; \text{avg. \# of characters per word in } S$$

LM: an Example

- Training data: <s> <s> He can buy the can of soda .

LM: an Example

- Training data: <s> <s> He can buy the can of soda .
- <u>Unigram model (n=8):</u>
    - $p_1(\text{He}) = p_1(\text{buy}) = p_1(\text{the}) = p_1(\text{of}) = p_1(\text{soda}) = p_1(.) = 0.125$,
      $p_1(\text{can}) = 0.25$

LM: an Example

- Training data: <s> <s> He can buy the can of soda .
- Unigram model (n=8):
  - $p_1(\text{He}) = p_1(\text{buy}) = p_1(\text{the}) = p_1(\text{of}) = p_1(\text{soda}) = p_1(.) = 0.125$, $p_1(\text{can}) = 0.25$
- Bigram model:
  - $p_2(\text{He}|\text{<s>}) = 1$, $p_2(\text{can}|\text{He}) = 1$, $p_2(\text{buy}|\text{can}) = 0.5$, $p_2(\text{of}|\text{can}) = 0.5$, $p_2(\text{the}|\text{buy}) = 1, \ldots$

LM: an Example

- Training data: $\langle s \rangle \langle s \rangle$ He can buy the can of soda .
- Unigram model (n=8):
  - p_1(He) = p_1(buy) = p_1(the) = p_1(of) = p_1(soda) = p_1(.) = 0.125, p_1(can) = 0.25
- Bigram model:
  - $p_2(\text{He}|\langle s \rangle) = 1$, $p_2(\text{can}|\text{He}) = 1$, $p_2(\text{buy}|\text{can}) = 0.5$, $p_2(\text{of}|\text{can}) = 0.5$, $p_2(\text{the}|\text{buy}) = 1$,...
- Trigram model:
  - $p_3(\text{He}|\langle s \rangle,\langle s \rangle) = 1$, $p_3(\text{can}|\langle s \rangle,\text{He}) = 1$, $p_3(\text{buy}|\text{He,can}) = 1$, $p_3(\text{of}|\text{the,can}) = 1$, ..., $p_3(.|\text{of,soda}) = 1$

LM: an Example

- Training data: <s> <s> He can buy the can of soda .
- Unigram model (n=8):
  - $p_1(\text{He}) = p_1(\text{buy}) = p_1(\text{the}) = p_1(\text{of}) = p_1(\text{soda}) = p_1(.) = 0.125$, $p_1(\text{can}) = 0.25$
- Bigram model:
  - $p_2(\text{He}|\text{<s>}) = 1$, $p_2(\text{can}|\text{He}) = 1$, $p_2(\text{buy}|\text{can}) = 0.5$, $p_2(\text{of}|\text{can}) = 0.5$, $p_2(\text{the}|\text{buy}) = 1, \ldots$
- Trigram model:
  - $p_3(\text{He}|\text{<s>},\text{<s>}) = 1$, $p_3(\text{can}|\text{<s>},\text{He}) = 1$, $p_3(\text{buy}|\text{He},\text{can}) = 1$, $p_3(\text{of}|\text{the},\text{can}) = 1, \ldots, p_3(.|\text{of},\text{soda}) = 1$
- Entropy: $H(p_1) = 2.75$, $H(p_2) = 0.25$, $H(p_3) = 0$ ← *Great?!*

LM: an Example (The Problem)

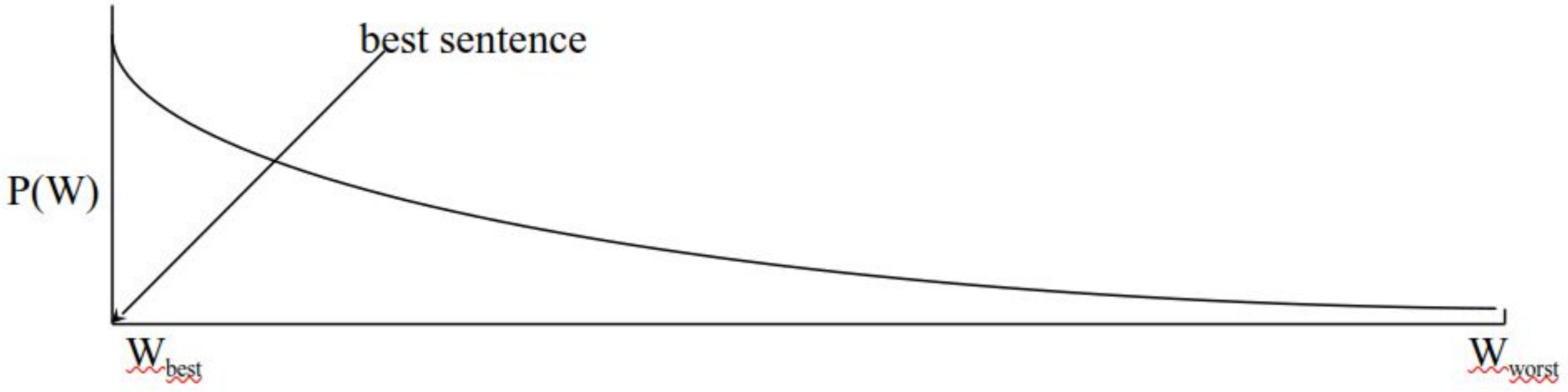
- Test data:

$$S = \text{<s> <s> It was the greatest buy of all.}$$

- Cross entropy:
    - 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 $p_1(.)$ are 0.
    - all bigram probabilities are 0.
    - all trigram probabilities are 0.
- We want:
    - to make all (theoretically possible) probabilities non-zero.

Real World Situation (from Lecture 1)

- Unable to specify set of grammatical sentences today using fixed "categorical" rules (maybe never, cf. arguments in M&S)
- Use statistical "model" based on **real world data** and care about the best sentence only (disregarding the "grammaticality" issue)



Moodle Quiz

Moodle Quiz

https://dl1.cuni.cz/course/view.php?id=18547