Language Models

Natural Language Processing

Daniel Ortiz Martínez, David Buchaca Prats

What is a Language Model

- A language model is a probabilistic model that assigns probabilities to text or strings
- Examples of problems that benefit from language models:
    - Spelling correction
    - Machine translation
    - Speech recognition
    - Handwriting recognition
    - Word suggestions in text editors

What is a Language Model

- Let us consider a toy corpus from https://en.wikipedia.org/wiki/This_Is_the_House_That_Jack_Built

This is the house that Jack built
This is the malt
That lay in the house that Jack built
This is the rat
That ate the malt
That lay in the house that Jack built
This is the cat
That killed the rat
That ate the malt
That lay in the house that Jack build

What is a Language Model

- We want to compute a bi-gram language model, which assumes

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

This is the house that Jack built
This is the malt
That lay in the house that Jack built
This is the rat
That ate the malt
That lay in the house that Jack built
This is the cat
That killed the rat
That ate the malt
That lay in the house that Jack build

What is a Language Model

- We want to compute a bi-gram language model, which assumes

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

This is the house that Jack built
This is the malt
That lay in the house that Jack built
This is the rat
That ate the malt
That lay in the house that Jack built
This is the cat
That killed the rat
That ate the malt
That lay in the house that Jack build

How could we estimate $p(w_i, w_j)$?

Markov Assumption

- N-gram language models make the Markov assumption
- Let us consider $x = (w_1, w_2, w_3, ..., w_N)$
- Using the chain rule of probabilities:

$$p(x) = p(w_1)p(w_2|w_1)p(w_3|w_2, w_1)...p(w_N|w_{N-1}, ..., w_2, w_1)$$

- A k-Markov assumption results in:

$$p(w_n|w_{n-1}, ..., w_1) = p(w_n|w_{n-1}, ..., w_{n-k+1})$$

Examples of Markov Assumptions

- If a k-Markov assumption is made

$$p(w_n|w_{n-1},...,w_1) = p(w_n|w_{n-1},...,w_{n-k+1})$$

- Example: if $k = 2$ and $N = 4$

$$p(w_4|w_3, w_2, w_1) = p(w_4|w_{n-1},...,w_{n-k+1}) = p(w_4|w_{4-1},...,w_{4-2+1}) = p(w_4|w_3)$$

- Example: $k = 3$ and $N = 4$

$$p(w_4|w_3, w_2, w_1) = p(w_4|w_{n-1},...,w_{n-k+1}) = p(w_4|w_{4-1},...,w_{4-3+1}) = p(w_4|w_3, w_2)$$

Examples of Markov Assumptions

- An expression of the form $p(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$ requires us to store the counts of pairs of words and single words
- Alternatively, the counts can be computed on the fly:

$$p(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})} = \frac{C(w_{n-1}w_n)}{\sum_{\tilde{w}} C(w_{n-1}\tilde{w})}$$

NOTE: The expression that marginalizes over $\tilde{w}$ has the advantage that needs less memory but requires summing over $\tilde{w}$ at runtime

What is a Language Model

- We want to compute a bi-gram language model, which assumes

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

This is the house that Jack built
This is the malt
That lay in the house that Jack built
This is the rat
That ate the malt
That lay in the house that Jack built
This is the cat
That killed the rat
That ate the malt
That lay in the house that Jack build

How could we estimate $p(w_i, w_j)$?
Divide the count of (w_i, w_j) in the corpus
by the count of w_j

What is a Language Model

- We want to compute a bi-gram language model, which assumes

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

This is the house that Jack built
This is the malt
That lay in the house that Jack built
This is the rat
That ate the malt
That lay in the house that Jack built
This is the cat
That killed the rat
That ate the malt
That lay in the house that Jack build

How could we estimate $p(w_i, w_j)$?
Divide the count of (w_i, w_j) in the corpus by the count of w_j

$$p(\text{house}|\text{the}) = \frac{\text{the house}}{\text{the}} = ?$$

What is a Language Model

- We want to compute a bi-gram language model, which assumes

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

This is the house that Jack built

This is the malt

That lay in the house that Jack built

This is the rat

That ate the malt

That lay in the house that Jack built

This is the cat

That killed the rat

That ate the malt

That lay in the house that Jack build

How could we estimate $p(w_i, w_j)$?

Divide the count of (w_i, w_j) in the corpus

by the count of w_j

$$p(\text{house}|\text{the}) = \frac{\text{the house}}{\text{the}} = ?$$

What is a Language Model

- We want to compute a bi-gram language model, which assumes

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

This is the house that Jack built

This is the malt

That lay in the house that Jack built

This is the rat

That ate the malt

That lay in the house that Jack built

This is the cat

That killed the rat

That ate the malt

That lay in the house that Jack build

How could we estimate $p(w_i, w_j)$?

Divide the count of (w_i, w_j) in the corpus

by the count of w_j

$$p(\text{house}|\text{the}) = \frac{\text{the house}}{\text{the}} = \frac{4}{10}$$

What is a Language Model

- We want to compute a four-gram language model, which assumes

$$p(w_n|w_{n-1}, ..., w_1) \approx p(w_n, w_{n-1}, w_{n-2}, w_{n-3})/p(w_{n-1}, w_{n-2}, w_{n-3})$$

This is the house that Jack built

This is the malt

That lay in the house that Jack built

This is the rat

That ate the malt

That lay in the house that Jack built

This is the cat

That killed the rat

That ate the malt

That lay in the house that Jack build

$$p(\text{house}|\text{the}\,,\text{is}\,,\text{this}) = \frac{\text{this is the house}}{\text{this is the}} = ?$$

What is a Language Model

- We want to compute a four-gram language model, which assumes

$$p(w_n|w_{n-1},...,w_1) \approx p(w_n, w_{n-1}, w_{n-2}, w_{n-3})/p(w_{n-1}, w_{n-2}, w_{n-3})$$

This is the house that Jack built

This is the malt

That lay in the house that Jack built

This is the rat

That ate the malt

That lay in the house that Jack built

This is the cat

That killed the rat

That ate the malt

That lay in the house that Jack build

$$p(\text{house}|\text{the}\,,\text{is}\,,\text{this}) = \frac{\text{this is the house}}{\text{this is the}} =?$$

What is a Language Model

- We want to compute a four-gram language model, which assumes

$$p(w_n|w_{n-1}, ..., w_1) \approx p(w_n, w_{n-1}, w_{n-2}, w_{n-3})/p(w_{n-1}, w_{n-2}, w_{n-3})$$

This is the house that Jack built
This is the malt
That lay in the house that Jack built
This is the rat
That ate the malt
That lay in the house that Jack built
This is the cat
That killed the rat
That ate the malt
That lay in the house that Jack build

$$p(\text{house}|\text{the}, \text{is}, \text{this}) = \frac{\text{this is the house}}{\text{this is the}} = \frac{1}{4}$$

Language Model with <s> and </s>

- Symbols <s> and </s> denote the start and stop of a sentence
- Using this special words will affect the model probabilities
- Consider the corpus:
  <s> This is the malt </s>
  <s> That lay in the house that Jack built </s>
- $p(\text{this is the house})$ for 2-gram language model without <s> and </s>:

$$p(\text{this is the house}) = p(\text{this})p(\text{is}|\text{this})p(\text{the}|\text{is})p(\text{house}|\text{the}) = \frac{1}{12} \cdot \frac{1}{1} \cdot \frac{1}{1} \cdot \frac{1}{2} = \frac{1}{24}$$

- $p(\text{this is the house})$ for 2-gram language model with <s> and </s>:

$$p(<\text{s}> \text{this is the house} </\text{s}>) = p(\text{this}|<\text{s}>)p(\text{is}|\text{this})p(\text{the}|\text{is})p(\text{house}|\text{the})p(</\text{s}>|\text{house}) = ?$$

Language Model with <s> and </s>

- Symbols <s> and </s> denote the start and stop of a sentence
- Using this special words will affect the model probabilities
- Consider the corpus:
  <s> This is the malt </s>
  <s> That lay in the house that Jack built </s>
- $p(\text{this is the house})$ for 2-gram language model without <s> and </s>:

$$p(\text{this is the house}) = p(\text{this})p(\text{is}|\text{this})p(\text{the}|\text{is})p(\text{house}|\text{the}) = \frac{1}{12} \cdot \frac{1}{1} \cdot \frac{1}{1} \cdot \frac{1}{2} = \frac{1}{24}$$

- $p(\text{this is the house})$ for 2-gram language model with <s> and </s>:

$$p(<\text{s}> \text{this is the house} <\text{/s}>) = p(\text{this}|<\text{s}>)p(\text{is}|\text{this})p(\text{the}|\text{is})p(\text{house}|\text{the})p(<\text{/s}>|\text{house}) = \frac{1}{2} \cdot \frac{1}{1} \cdot \frac{1}{1} \cdot \frac{1}{2} \cdot 0 = 0$$

Zipf's Law

- The Zipf's law is an empirical law that may hold when a list of measured values is sorted in decreasing order
- The best known instance applies to the frequency table of words in a natural language text

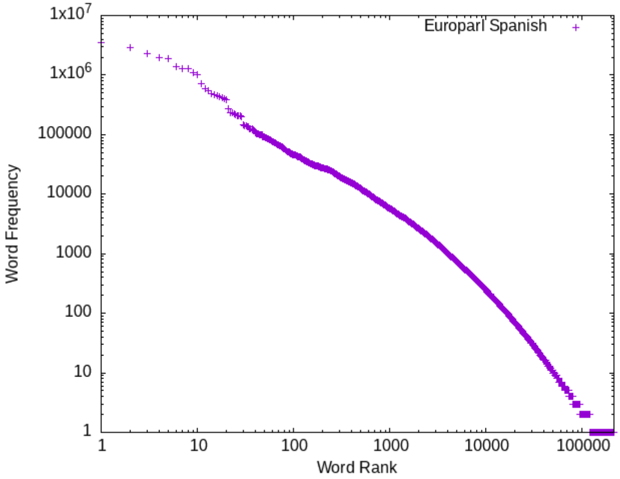
$$\text{word frequency} \propto \frac{1}{\text{word rank}}$$

- Word rank refers to the position of a word within a frequency-ordered list of words in a text

Zipf's Law

- A Zipf-like distribution is obtained when representing word ranks against word frequencies
- In a Zipfian distribution, the most common word occurs approximately twice as often as the next common one, three times as often as the third most common, and so on

Zipf's Law: Example for Europarl Corpus*

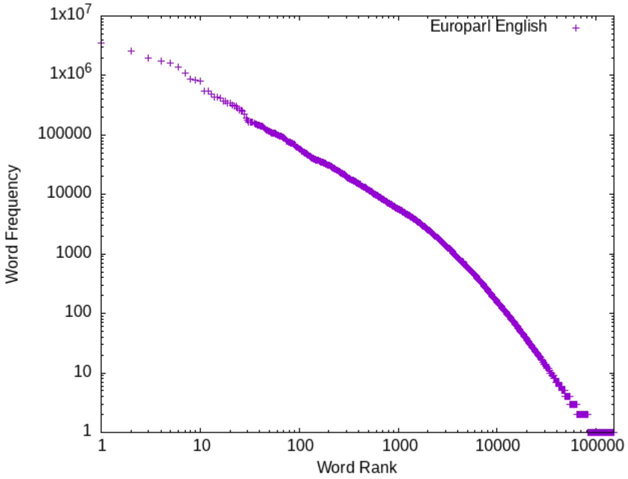


*See this link for more information

Zipf's Law: Exercise

- **Exercise**: generate the Zipf plot for the English part of the Europarl corpus (the raw data can be downloaded here)

Zipf's Law: Exercise



Zipf's Law and Unseen *n*-grams

- The Zipf's law determines that in a given text, there will be a few words that are very common and many words that are very uncommon
- This means that estimating accurate probabilities for rare words is challenging due to limited data

Dealing with Unseen *n*-grams: Smoothing

- Smoothing techniques are a key element in *n*-gram language models
- If we plot the probabilities of the events described by the model, smoothing techniques have the goal of making the distribution less sharp (or smoother)
- Smoothing tries to correct the problems derived of data scarcity and Zipf's law: a few common words will have too much probability and many words will have too few (or zero) probability

Dealing with Unseen *n*-grams: Laplace Smoothing

- We want to allow unseen grams to have non zero probability
- A common technique is called *Laplace smoothing* or *Add 1 smoothing*

$$p_{\mathrm{add}_1}(w_i|w_j) = \frac{c(w_j w_i)+1}{c(w_j)+|V|} \quad p_{\mathrm{add}_1}(w_i) = \frac{c(w_i)+1}{c()+|V|} = \frac{c(w_i)+1}{\#\mathrm{tokens}+|V|}$$

- Consider the corpus:
  <s> This is the malt </s>
  <s> That lay in the house that Jack built </s>
- With $V = \{$This,is,the,malt,That,lay,in,house,that,Jack,build,<s>,</s>$\}$

$$p(<\mathrm{s}>\mathrm{this\ is\ the\ house}</\mathrm{s}>) = p(\mathrm{this}|<\mathrm{s}>)p(\mathrm{is}|\mathrm{this})p(\mathrm{the}|\mathrm{is})p(\mathrm{house}|\mathrm{the})p(</\mathrm{s}>|\mathrm{house}) =$$

Dealing with Unseen n-grams: Laplace Smoothing

- We want to allow unseen grams to have non zero probability
- A common technique is called *Laplace smoothing* or *Add 1 smoothing*

$$p_{\text{add}_1}(w_i|w_j) = \frac{c(w_j w_i)+1}{c(w_j)+|V|} \quad p_{\text{add}_1}(w_i) = \frac{c(w_i)+1}{c()+|V|} = \frac{c(w_i)+1}{\#\text{tokens}+|V|}$$

- Consider the corpus:
  <s> This is the malt </s>
  <s> That lay in the house that Jack built </s>
- With $V = \{$This,is,the,malt,That,lay,in,house,that,Jack,build,<s>,</s>$\}$

$$p(<\text{s}> \text{this is the house} </\text{s}>) = p(\text{this}|<\text{s}>)p(\text{is}|\text{this})p(\text{the}|\text{is})p(\text{house}|\text{the})p(</\text{s}>|\text{house}) =$$
$$\frac{1+1}{2+13}\cdot\frac{1+1}{1+13}\cdot\frac{1+1}{1+13}\cdot\frac{1+1}{2+13}\cdot\frac{0+1}{1+13}$$

Dealing with Unseen n-grams: Lidstone Smoothing

- Laplace smoothing is a particular case of a more general technique: *Lidstone Smoothing*

$$p_{\text{lidstone}}(w_i|w_j) = \frac{c(w_j w_i) + \epsilon}{c(w_j) + |V| \cdot \epsilon} \quad p_{\text{lidstone}}(w_i) = \frac{c(w_i) + \epsilon}{c() + |V| \cdot \epsilon} = \frac{c(w_i) + \epsilon}{\#\text{tokens} + |V| \cdot \epsilon}$$

- Consider the corpus:
  <s> This is the malt </s>
  <s> That lay in the house that Jack built </s>

- With $V = \{$This,is,the,malt,That,lay,in,house,that,Jack,build,<s>,</s>$\}$

$$p(<\text{s}>\text{ this is the house }</\text{s}>) = p(\text{this}|<\text{s}>)p(\text{is}|\text{this})p(\text{the}|\text{is})p(\text{house}|\text{the})p(</\text{s}>|\text{house}) =$$

Dealing with Unseen *n*-grams: Lidstone Smoothing

- Laplace smoothing is a particular case of a more general technique: *Lidstone Smoothing*

$$p_{\text{lidstone}}(w_i|w_j) = \frac{c(w_j w_i) + \epsilon}{c(w_j) + |V| \cdot \epsilon} \quad p_{\text{lidstone}}(w_i) = \frac{c(w_i) + \epsilon}{c() + |V| \cdot \epsilon} = \frac{c(w_i) + \epsilon}{\#\text{tokens} + |V| \cdot \epsilon}$$

- Consider the corpus:

  <s> This is the malt </s>

  <s> That lay in the house that Jack built </s>

- With $V = \{$This,is,the,malt,That,lay,in,house,that,Jack,build,<s>,</s>$\}$

$$p(<\text{s}>\text{ this is the house }</\text{s}>) = p(\text{this}|<\text{s}>)p(\text{is}|\text{this})p(\text{the}|\text{is})p(\text{house}|\text{the})p(</\text{s}>|\text{house}) =$$

$$\frac{1+0.5}{2+13\cdot 0.5} \cdot \frac{1+0.5}{1+13\cdot 0.5} \cdot \frac{1+0.5}{1+13\cdot 0.5} \cdot \frac{1+0.5}{2+13\cdot 0.5} \cdot \frac{0+1}{1+13\cdot 0.5} \quad \text{for } \epsilon = 0.5$$

Backoff Models

- Lidstone smoothing is based on *discounting* probability mass from observed n-grams and redistributing it
    - the discounting is done by increasing the denominator of the relative frequency estimates

- An alternative smoothing technique is *backoff*
    - if a particular n-gram has not been seen, use $(n-1)$-grams

Backoff Models

- Equation for Katz's backoff model[†]:

$$P_{\mathrm{bo}}(w_i|w_{i-1}\dots w_{i-n+1}) = \begin{cases} d_{w_{i-n+1}\cdots w_i} \frac{C(w_{i-n+1}\cdots w_i)}{C(w_{i-n+1}\cdots w_{i-1})} & \text{if}\, C(w_{i-n+1}\dots w_i) > k \\ \alpha_{w_{i-n+1}\cdots w_{i-1}} P_{\mathrm{bo}(w_i|w_{i-1}\cdots w_{i-n+2})} & \text{otherwise} \end{cases}$$

where:

  - k is typically defined as zero
  - d is the discounting amount using the so-called *Good-Turing* estimation
  - α is another discounting amount called *backoff weight*

[†]Follow this link for a detailed explanation

Interpolation

- Interpolation is another way of combining different order *n*-gram models
- Instead of choosing probabilities for a particular *n*-gram order, interpolation takes the weighted average of multiple orders
- Example of interpolated trigram model[‡]:

$$\begin{aligned} p_{\text{interpolation}(w_n|w_{n-1},w_{n-2})} = & \lambda_3 p_3^*(w_n|w_{n-1}, w_{n-2}) \\ & + \lambda_2 p_2^*(w_n|w_{n-1}) \\ & + \lambda_1 p_1^*(w_n) \end{aligned}$$

- Interpolation weights can be obtained by means of the EM algorithm

[‡]where p_n^* is the unsmoothed probability of an *n*-gram language model and λ_n is the weight assigned to this model

Kneser-Ney Smoothing

- Kneser-Ney (KN) smoothing is widely considered the most effective smoothing technique for *n*-gram language models
- KN smoothing is a discounting technique that pays special attention to the number of different words that precede a given one
- Example: consider the bigram “San Francisco”
    - If the training text contains the bigram many times, then the unigram “Francisco” is going to have a high frequency too
    - A standard unigram model would make skewed predictions
    - KN smoothing will consider the frequency of the unigram in relation to possible words preceding it (“Francisco” will have a low probability)

Kneser-Ney Smoothing

- Equation for bigram probabilities:

$$p_{\mathrm{KN}}(w_i|w_{i-1}) = \frac{\max\left(c(w_{i-1}, w_i) - \delta, 0\right)}{\sum_{w'} c(w_{i-1}, w')} + \lambda_{w_{i-1}} p_{\mathrm{KN}(w_i)}$$

  where:

  - $\lambda_{w_{i-1}}$ is a normalizing constant
  - $p_{\mathrm{KN}}(w_i)$ is the unigram probability:

$$p_{\mathrm{KN}}(w_i) = \frac{|\{w : 0 < c(w, w_i)\}|}{\sum_{u \in V} |\{w' : 0 < c(u, w')\}|}$$

  - $|\{w : 0 < c(w, w')\}|$ is the number of different words that precede w'

Scalability Issues of *n*-gram Language Models

- Storing all counts for n-grams might require a lot of memory if N is big
- 140 GB of RAM for language model described in https://kheafield.com/papers/edinburgh/estimate_paper.pdf
- Neural Networks offer an alternative where RAM requirements scale with the number of words

Exercise

- Given the corpus
  <s> I am Sam </s>
  <s> Sam I am </s>
  <s> I do not like green eggs and ham </s>
- Compute

$$
\begin{aligned}
P(\mathrm{I}|<\mathrm{s}>) &= \\
P(\mathrm{Sam}|<\mathrm{s}>) &= \\
P(\mathrm{am}|\mathrm{I}) &= \\
P(</\mathrm{s}>|\mathrm{Sam}) &= \\
P(\mathrm{Sam}|\mathrm{am}) &= \\
P(\mathrm{do}|\mathrm{I}) &=
\end{aligned}
$$

Exercise

- Given the corpus
  <s> I am Sam </s>
  <s> Sam I am </s>
  <s> I do not like green eggs and ham </s>
- Compute

$$\begin{aligned} P(\mathrm{I}|<\mathrm{s}>) &= 2/3 \\ P(\mathrm{Sam}|<\mathrm{s}>) &= 1/3 \\ P(\mathrm{am}|\mathrm{I}) &= 2/3 \\ P(</\mathrm{s}>|\mathrm{Sam}) &= 1/2 \\ P(\mathrm{Sam}|\mathrm{am}) &= 1/2 \\ P(\mathrm{do}|\mathrm{I}) &= 1/3 \end{aligned}$$

Numerical Stability with n-gram Models

- When computing the probability for a long sentence you end up multiplying a lot of probabilities over the n-grams. Each product of two small numbers becomes an even smaller number

$$p(<\mathrm{s}> \text{ this is the house } </\mathrm{s}>) = p(\text{this}|<\mathrm{s}>)p(\text{is}|\text{this})p(\text{the}|\text{is})p(\text{house}|\text{the})p(</\mathrm{s}>|\text{house})$$

- This may result in numerical underflow

$$0.0001 \cdot 0.0003 \cdot 0.0003 \cdot 0.0003 \cdot 0.0005 = 1.349e - 18$$

Computing in Log Domain

- Computing in log domain allows us to deal with products of probabilities in a smart approach that can avoid numerical problems
- If a number a is positive then $a = \exp(\log(a)) = \log(\exp(a))$
- Therefore we can compute:

$$p_1 \cdot p_2 \cdot p_3 = \exp(\log(p_1 \cdot p_2 \cdot p_3)) = \exp(\log(p_1) + \log(p_2) + \log(p_3))$$

Computing in Log Domain

- If the previous equality is an equality why do we care?
  - Because summing $\log(p_i)$ will not result to numerical underflow but multiplying the p_i might.
- In practice, we will only use probabilities if we need to report them at the end
- The rest of the time, all calculations and storage are made in log space

Evaluating Language Models

- One of the most common metrics to evaluate a language model is perplexity
- The perplexity of a language model on a test set is defined as the inverse probability of the test set, normalized by the number of words
- The higher the probability the lower the perplexity

Evaluating Language Models

- For a test set $X = w_1, ..., w_N$ the perplexity of a model h on X is:

$$\mathrm{PP}(h; X) = P_h(w_1, ..., w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P_h(w_1, ..., w_N)}}$$

- If we use the chain rule to compute $P_h(w_1, ..., w_N)$ we have

$$\mathrm{PP}(h; X) = \sqrt[N]{\frac{1}{P_h(w_1, ..., w_N)}} = \sqrt[N]{\frac{1}{\prod_{n=1}^{N} P_h(w_i | w_{i-1}, ..., w_1)}} = \sqrt[N]{\prod_{n=1}^{N} \frac{1}{P_h(w_i | w_{i-1}, ..., w_1)}}$$

Evaluating Language Models

- Note that in the definition of perplexity we used $X = w_1, ..., w_N$
- Here we mean the concatenation of all the lines in the test set. Since the test set will contain many sentence boundaries we need to include them in the probability computation
- Given the test set
  <s> I am Sam </s>
  <s> Sam I am </s>
  <s> I do not like green eggs and ham </s>
- We represent the whole test set in a single line:
  <s> I am Sam </s> <s> Sam I am </s> <s> I do not like green eggs and ham </s>