Chapter NLP:III

III. Text Models

- Text Preprocessing
- Text Representation
- Text Similarity
- Text Classification
- Language Modeling
- Sequence Modeling

Language Modeling

Definition 1 (Language Model)

A language model is a probability distribution $P(w_1, ..., w_n)$ over all sequences of tokens $\langle w_1, ..., w_n \rangle \in V^*$ supported by a fixed-size vocabulary $\mathbf{V}$, with:

1. $\forall \langle w_1, ..., w_n \rangle \in V^* : P(w_1, ..., w_n) \geq 0$, and
2. $$\sum_{\langle w_1, ..., w_n \rangle \in V^*} P(w_1, ..., w_n) = 1.$$

Definition 2 (Language Modeling)

Language modeling is the task of estimating a probability function $P(w_n | w_1, ..., w_{n-1})$.

- Alternative Fomulation: Predict the next token w_n in a sequence given the history $w_1, ..., w_{n-1}$.
- This can be seen as a classification task.

Remarks:

- Modeling $P(w_n|w_1, ..., w_{n-1})$ is sometimes called autoregressive or causal language modeling.
- Consider the duality of the definitions: Language models can predict $P(w_n|w_1, ..., w_{n-1})$, since $w_n = \max_{w_i \in V} P(w_1, ..., w_{n-1}, w_i)$ is known.

Language Modeling

Implications from the Definition

Likelihood estimation

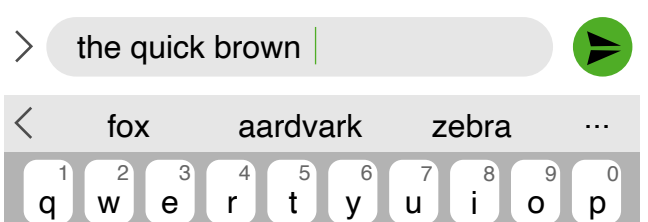
- Given two different sequences, a language model can decide which sequence is more likely.
  - In spell- and grammarchecking, errors can be detected when sequences become highly unlikely.
- Language model predictions depend on the sequences used to contruct the model. So models constructed from different sources predict different probabilities for the same sequence.
  - In language identification, a phrase in an unknown languge can be scored by language models constructed from texts in different languages.
  - In information retrieval, a query can be scored by a language model constructed from each document to be ranked.

Language Modeling

Implications from the Definition

Prediction and Generation

- If a model estimates $P(w_n|w_1, ..., w_{n-1})$, it estimates $P(w_{n+1}|w_1, ..., w_n)$ as well.
- Thus, language models can generate text of arbitraty length, given a prompt of $w_1, ..., w_{n-1}$



Language Modeling

Language Model Estimation

Language models can be estimated from observations in a corpus via maximum likelihood estimation (MLE):

$$P_{MLE}(w_1, ..., w_n) = \frac{C(w_1, ..., w_n)}{N}$$

$$P_{MLE}(w_n|w_1, ..., w_{n-1}) = \frac{C(w_1, ..., w_{n-1}, w_n)}{\sum_w C(w_1, ..., w_{n-1}, w)}$$

$$= \frac{C(w_1, ..., w_n)}{C(w_1, ..., w_{n-1})}$$

1. Segment the corpus into all possible n-grams $w_1, ..., w_n$.
2. Count the occurrences $C(w_1, ..., w_n)$ of each unique n-gram.
3. Get the probabilities $P(w_1, ..., w_n)$ by dividing by the total (N).

Language Modeling

Language Model Estimation

Language models can be estimated from observations in a corpus via maximum likelihood estimation (MLE):

$$P_{MLE}(w_1, ..., w_n) = \frac{C(w_1, ..., w_n)}{N}$$

$$P_{MLE}(w_n | w_1, ..., w_{n-1}) = \frac{C(w_1, ..., w_{n-1}, w_n)}{C(w_1, ..., w_{n-1})}$$

`the quick brown fox jumps over the lazy dog`

$w_1, ..., w_{n-1}, w_n$	$C(w_1, ..., w_{n-1})$	$C(w_1, ..., w_n)$	$P(w_n \| w_1, ..., w_{n-1})$	$P(w_1, ..., w_n)$
`the quick`	2	1	1/2 = 0.5	1/45
`the quick brown`	1	1	1/1 = 1	1/45
`the quick brown fox`	1	1	1/1 = 1	1/45

What are the problems with this model?

Language Modeling

Language Model Estimation

Chain rule of probability (see Bayes' theorem):

$$P(X_1...X_n) = P(X_1)P(X_2|X_1)P(X_3|P_{1,2})...P(X_n|X_{1,...,n-1}) = \prod_{k=1}^{n} P(X_k|X_{1,...,k-1})$$

Chain rule applied to language:

$$P(w_1, ..., w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)...P(w_n|w_1, ..., w_{n-1}) = \prod_{k=1}^{n} P(w_k|w_{1,...,k-1})$$

Example:

$$\begin{aligned} P(w_1, ..., w_n) =& P(\text{the quick brown fox jumps over the lazy dog}) \\ =& P(\text{the}) \cdot \\ & P(\text{quick}|\text{the}) \cdot \\ & P(\text{brown}|\text{the quick}) \cdot \\ & ... \cdot \\ & P(\text{dog}|\text{the quick brown fox jumps over the lazy}) \end{aligned}$$

What are the problems with this model?

Language Modeling

Language Model Estimation

We can reduce the length of the condition with a markov assumption: The language model $P(w_n|w_1, ..., w_{n-1})$ is a memoryless stochastic process, so the probability of observing w_n as the next word depends only on the current observed word w_{n-1}:

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

Chain rule applied to language under a markov assumption:

$$P(w_1, ..., w_n) = P(w_1|\text{<s>})P(w_2|w_1)P(w_3|w_2)...P(w_n|w_{n-1}) = \prod_{k=1}^{n} P(w_k|w_{k-1})$$

Example:

$$\begin{aligned} P(w_1, ..., w_n) =& P(\text{the quick brown fox jumps over the lazy dog}) \\ =& P(\text{the}|\text{<s>}) \cdot \\ & P(\text{quick}|\text{the}) \cdot \\ & P(\text{brown}|\text{quick}) \cdot \\ & ... \cdot \\ & P(\text{dog}|\text{lazy}) \end{aligned}$$

Remarks:

- The condition for Bayes' theorem is that the predicted events are mutually exclusive. Hence, we must assume that words in a sequence are independent, which is not true.
- The language model parameters can also be estimated through other methods, like Bayes' rule, but MLE is the most common.
- The `</s>` token is neccessary to estimate one distribution over all seqences as in the definition. If `</s>` is left, the bigram model would estimate one distribution for all sequences of the same length.
- The probabilities of long sequences become very small. This requires arbitrary floating-point precision, which is often a computational problem. This can be avoided using log probabilities, which turns the multiplication into a sum of logarithms:

$$\prod_{k=1}^{n} P(w_k|w_{1,\ldots,k-1}) = \exp(\sum_{k=1}^{n} log(P(w_k|w_{1,\ldots,k-1})))$$

- If a sequence has a token that is not in the vocabulary of the LM, the model can not estimate the sequences' probability. This can be solved by reducing the vocuabulary while fitting the model and replacing all out-of-vocabulary words with a special token `<unk>` .

Language Modeling

Bigram Model

The language model under a first-order markov assumption is called a bigram model. The paramete $P(w_n|w_{n-1})$ can be estimated via MLE:

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_{w_k \in V} C(w_{n-1}w_k)} = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

- The first parameter $P(w_1)$ is not explained by this estimation.
- This is usually avoided by adding a token `<s>` to the beginning of the sequence and estimating $P(w_1|\text{<s>})$ as the first probability.
- A token `</s>` is added to the end of each sequence.

Language Modeling

Bigram Model: Example

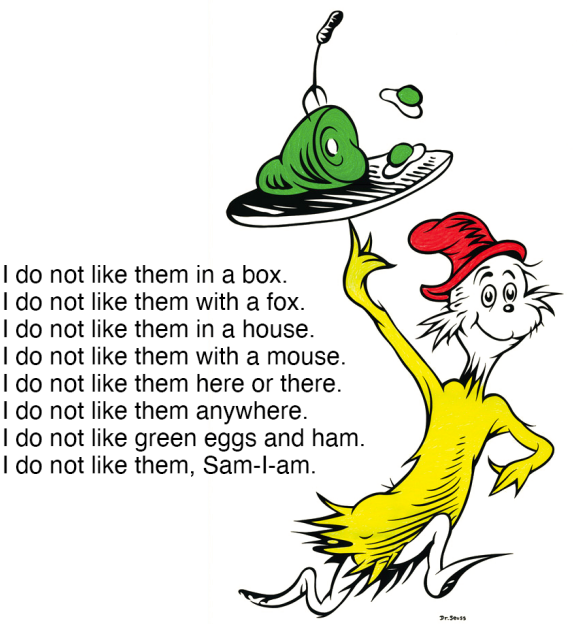
<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

$$P(\texttt{I}|\texttt{<s>}) = \frac{C(w_{n-1}w_n) = 2}{C(w_{n-1})}$$

$$P(\texttt{Sam}|\texttt{am}) =$$

$$P(\texttt{do}|\texttt{I}) =$$

	<s>	I	am	Sam	do	</s>	...
<s>							
I							
am							
Sam							
do							
...							



Language Modeling

Bigram Model: Example

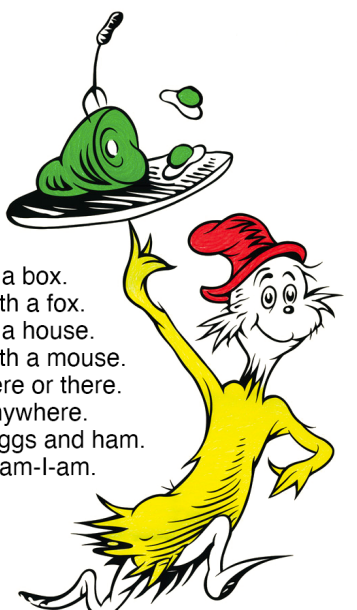
<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

$$P(\texttt{I}|\texttt{<s>}) = \frac{2}{C(w_{n-1}) = 3} = 0.67$$

$$P(\texttt{Sam}|\texttt{am}) =$$

$$P(\texttt{do}|\texttt{I}) =$$

	<s>	I	am	Sam	do	</s>	...
<s>		0.67					
I							
am							
Sam							
do							
...							



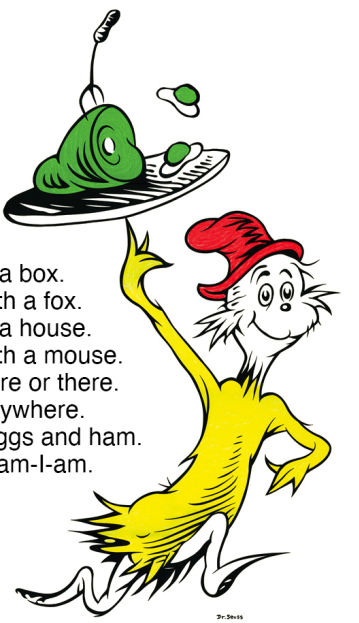
Language Modeling

Bigram Model: Example

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

$$P(\texttt{I}|\texttt{<s>}) = 0.67$$
$$P(\texttt{Sam}|\texttt{am}) = 0.5$$
$$P(\texttt{do}|\texttt{I}) =$$

	<s>	I	am	Sam	do	</s>	...
<s>		0.67					
I							
am				0.5			
Sam							
do							
...							



Language Modeling

Bigram Model: Example

```
<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>
```

$$P(\texttt{I}|\texttt{<s>}) = 2/3 = 0.67$$
$$P(\texttt{Sam}|\texttt{am}) = 1/2 = 0.50$$
$$P(\texttt{do}|\texttt{I}) = 1/3 = 0.33$$

	<s>	I	am	Sam	do	</s>	...
<s>	0	0.67	0	0.34	0	0	
I	0	0	0.67	0	0.34	0	
am	0	0	0	0.5	0	0.5	
Sam	0	0.5	0	0	0	0.5	
do	0	1	0	0	0	0	
...							

Language Modeling

The N-gram Model

The Bigram model can be generalized to arbitrary N-grams by relaxing the markov assumption to higher orders.

Generalized chain rule:

$$P(w_n|w_1, ..., w_{n-1}) \approx P(w_n|w_{n-N+1}, ..., w_{n-1})$$
$$= \prod_{k=1}^{n} P(w_k|w_{k-N+1}, ..., w_{k-1})$$

Generalized MSE:

$$P(w_n|w_{n-N+1}, ..., w_{n-1}) = \frac{C(w_{n-N+1}, ..., w_{n-1}, w_n)}{C(w_{n-N+1}, ..., w_{n-1})}$$

Bigram	am	I	do	Sam	not
<s>	0	.67	0	.34	0
I	.67	0	.34	0	0
am	0	0	0	.5	0
...					

Trigram	am	I	do	Sam	not
<s> I	.5	0	.5	0	0
<s> Sam	0	1	0	0	0
I am	0	1	0	.5	0
...					

4-gram	am	I	do	Sam	not
<s> I am	0	0	0	1	0
<s> Sam I	1	0	0	0	0
<s> I do	0	0	0	0	1
...					

Language Modeling

Improving the N-gram Model: Smoothing

The introduced n-gram model has two central problems:

1. We can not compute the probability if the denominator is zero.

$$P(\text{eggs}|\text{I like green}) = \frac{C(\text{I like green eggs})}{C(\text{I like green})}$$

2. The probability of the sequence is zero because the numerator is zero.

$$P(\text{ham}|\text{I do not like green}) = \frac{C(\text{I do not like green ham})}{C(\text{I do not like green})}$$

These problems can be avoided by redistributing probability mass from observed to unobserved events. This is called smoothing or discounting.

Language Modeling

Denominator Smoothing: Stupid Backoff

Idea: Consult a less specific model.

$$P(w_n|w_{n-k}, ..., w_{n-1}) \approx \cdots \approx P(w_n|w_{n-2}, w_{n-1}) \approx P(w_n|w_{n-1})$$

- Backoff progressively reduces the history, down to unigrams.

$$P_{BO}(w_n|w_{n-k}, ..., w_{n-1}) = \begin{cases} P(w_n|w_{n-k}, \ldots, w_{n-1}) & \text{if } C(w_{n-k}, \ldots, w_{n-1}) > \lambda \\ P_{BO}(w_n|w_{n-k+1}, \ldots, w_{n-1}) & \text{otherwise} \end{cases}$$

- The parameter λ is a lower bound to the frequency of the observation of the more specific model. This can either be set or estimated through evaluation (see Perplexity). Higher values require more support which can make the model more robust.

Language Modeling

Denominator Smoothing: Linear Interpolation

Idea: Use $(n-1)$-gram probabilities to smooth n-gram probabilities

- **Trigram** combination of the sentence does not occur in the training corpus
  - Exploiting the information about two-word form combination
- **Bigram** combination of the sentence does not occur in the training corpus
  - Exploiting the information about the single word form

Implementation: Utilization of all three sources of information (tri-, bi- and unigrams).

- But different weighting of the sources, as trigram is more informative than bigram etc.

Language Modeling

Denominator Smoothing: Linear Interpolation

Idea: Use $(n-1)$-gram probabilities to smooth n-gram probabilities

Instead of:

$$P(w_n|w_1, w_2, \ldots, w_{n-1}) \approx \frac{C(w_{n-2}, w_{n-1}, w_n)}{C(w_{n-2}, w_{n-1})}$$

we set:

$$P(w_n|w_1, w_2, \ldots, w_{n-1}) \approx \lambda_1 \frac{C(w_{n-2}, w_{n-1}, w_n)}{C(w_{n-2}, w_{n-1})} + \lambda_2 \frac{C(w_{n-1}, w_n)}{C(w_{n-1})} + \lambda_3 \frac{C(w_n)}{\sum_{k=1}^{N} C(w_k)}$$

with $\lambda \leq \lambda_i \leq 1$ and $\sum_i \lambda_i = 1$.

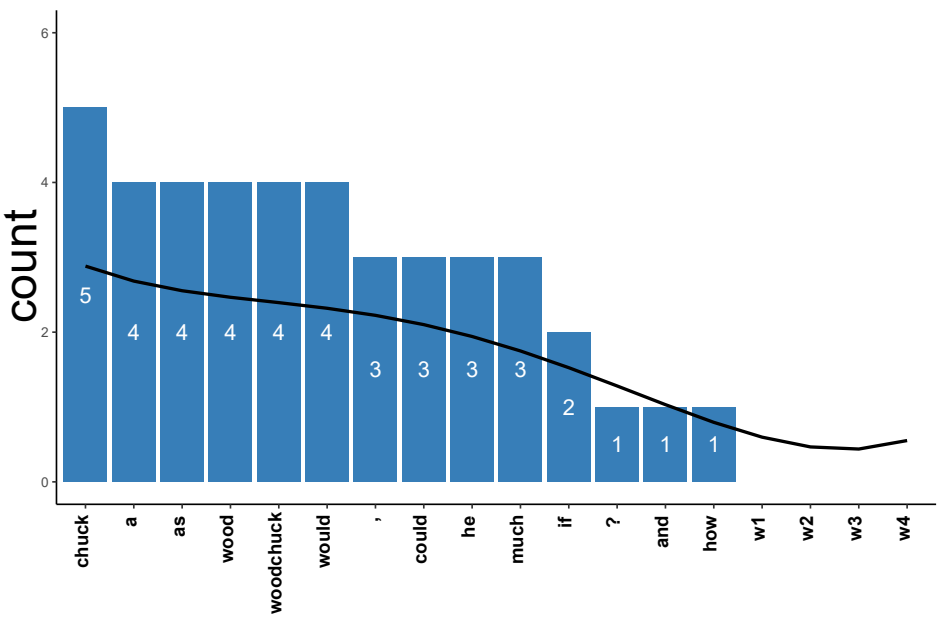
- λ can be set manually
- Additionally, automatic methods include Expectation Maximization (EM) algorithms, e.g. Hidden Markov Models

Language Modeling

Numerator Smoothing: Add-one (Laplace) smoothing

Example: Imagine the following text as as training input for a n-gram language model. Additionally image a situation where you want to obtain the probability of a sentence including the words w_1, w_2, w_3, w_4.

```
How much wood would a woodchuck chuck if a woodchuck could chuck wood?
He would chuck, he would, as much as he could, and chuck as much wood As
a woodchuck would if a woodchuck could chuck wood
```



- We normally get the probability of a word like:

$$P(W = w_i) = \frac{C(w_i)}{\sum_{j=1}^{m} C(w_j)}$$

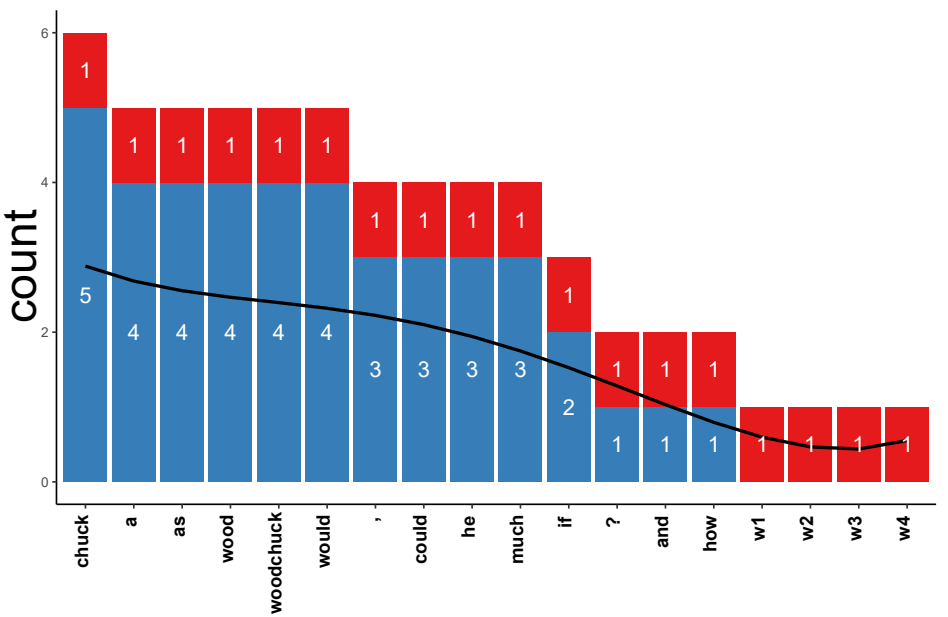
- In this model we assign 0 probability to w_1, w_2, w_3, w_4.

Language Modeling

Numerator Smoothing: Add-one (Laplace) smoothing

Example: Imagine the following text as as training input for a n-gram language model. Additionally image a situation where you want to obtain the probability of a sentence including the words w_1, w_2, w_3, w_4.

```
How much wood would a woodchuck chuck if a woodchuck could chuck wood?
He would chuck, he would, as much as he could, and chuck as much wood As
a woodchuck would if a woodchuck could chuck wood
```



- We can add 1 to each count:

$$P_{\text{add-1}}(W = w_i) = \frac{C(w_i) + 1}{\sum_{j=1}^{m} C(w_j) + V}$$

- In this model we assign some probability to w_1, w_2, w_3, w_4.

Language Modeling

Numerator Smoothing: Add-one (Laplace) smoothing

In general we could define the calculation of the probabilities of the n-grams as:

$$P_{\text{add-1}}(W = w_i) = \frac{C(w_i) + 1}{\sum_{j=1}^{m} C(w_j) + V}$$

$$P_{\text{add-1}}(w_n | w_{n-2}, w_{n-1}) = \frac{C(w_{n-2}, w_{n-1}, w_n) + 1}{C(w_{n-2}, w_{n-1}) + V}$$

We introduce pseudo counts like:

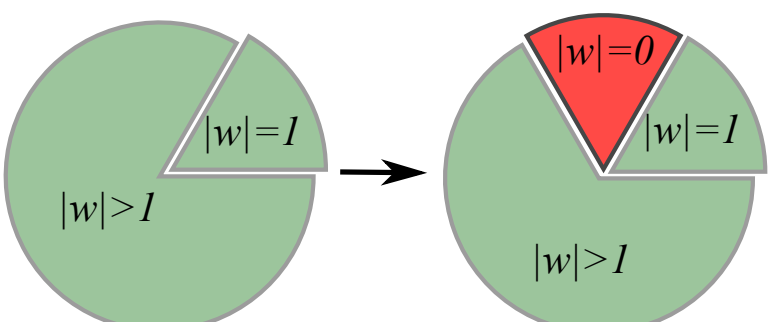
$$C(w_i)^* = (C(w_i) + 1) + \frac{\sum_{j=1}^{m} C(w_j)}{\sum_{j=1}^{m} C(w_j) + V}$$

where V is the total number of possible (N-1)-grams (i.e. the vocabulary size for a bigram model)

Language Modeling

Numerator Smoothing: Good-Turing smoothing

- In Add-one too much probability mass is moved! (In fact, we only use it for Naive Bayes with few unknown events)
- **Basic idea:** Use total frequency of events that occur only once to estimate how much mass to shift to unseen events.
- Let N_k be the number of N-grams that occur k times.
  - For bigrams, N_0 is the number of bigrams of count 0, N_1 is the number of bigrams with count 1, etc.



We revisit all counts by:

$$|w|^*_{k-1} = \frac{|w| \cdot N_k}{N_{k-1}}$$

All unseen words get the probability:

$$P(|w|_0) = \frac{N_1}{\sum_{\forall k} N_k}$$

Language Modeling

Numerator Smoothing: Kneser-Ney Smoothing

Observation: "Mercedes Benz" is frequent, but "Benz" only occurs after "Mercedes"

- Shannon game: `I want to drive my Mercedes to|Benz ...`
- `to` is much more common than `Mercedes`
- The unigram is useful exactly when we haven't seen this bigram!
- We do not use "How likely is w?"
- We introduce $P_{cont}(w)$: "How likely is w to appear as a novel continuation?"
  - For each word, count the number of bigram types it completes
  - Every bigram type was a novel continuation the first time it was seen

Idea: The unigram probability $P(w)$ should not depend on the frequency of w, but on the **number of contexts** in which w appears.

Language Modeling

Numerator Smoothing: Kneser-Ney Smoothing

We calculate how many times w appears as a novel continuation:

$$P_{cont}(w)|\{\forall w_{n-1} : |w_{n-1}, w_n| > 0\}|$$

and normalize by the total number of word bigram types:

$$P_{cont}(w) = \frac{|\{\forall w_{i-1} : |w_{i-1}, w| > 0\}|}{|\{\forall w_{j-1}, w_j : |w_{j-1}, w_j| > 0\}|}$$

Language Modeling

Numerator Smoothing: Kneser-Ney Smoothing

The Kneser-Ney Smoothing P_{KN} is then:

$$P_{KN}(w_i|w_{i-1}) = \frac{max(|w_{n-1}, w_n| - d, 0)}{|w_{n-1}|} + \lambda(w_{i-1}) \cdot P_{cont}(w_n)$$

λ is a normalizing constant for the probability mass which is discounted. Note, $|\{\forall w_n : |w_{n-1}, w_n| > 0\}|$ is the the number of word types that can follow w_{n-1}

$$\lambda(w_{i-1}) = \frac{d}{|w_{n-1}|}|\{\forall w_n : |w_{n-1}, w_n| > 0\}|$$

- The parameter d is a constant which denotes the discount value subtracted from the count of each n-gram, usually between 0 and 1.
- There is a recursion as a generalization for higher order ngrams see Jurafsky Book

Language Modeling

Evaluation: Perplexity

We can evaluate how good a language model is by how perplex (confused) it is about the test data.

- Perplexity is the inverse probability of the test data, normalized by the number of words:

$$\text{perplexity}(W) = P(w_1, \ldots, w_N)^{-\frac{1}{N}}$$

- Example for trigrams:

$$\text{perplexity}(W) = \sqrt[N]{\prod_{n=1}^{N} \frac{1}{P(w_n | w_{i-2}, w_{i-1})}}$$

- A low perplexity is desirable and means the model expected the sequences in the test data.

Remarks:

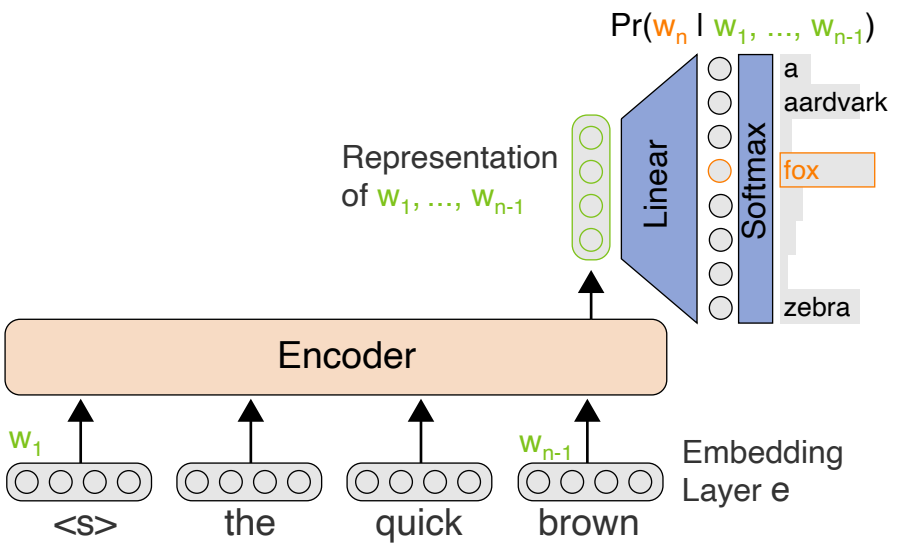
- Minimizing perplexity is the same as maximizing probability
- Perplexity usually goes down from unigram to trigram models.
- Smoothing, interpolation etc. raises probability (and so lowers the perplexity) of the model.
- The trigram formula directly follows from Bayes rule and the chain rule of probability (cf. NLP:III-157). The dividend and N-root follows from the inverse normalization.

Language Modeling

Neural Language Models

Intuition:

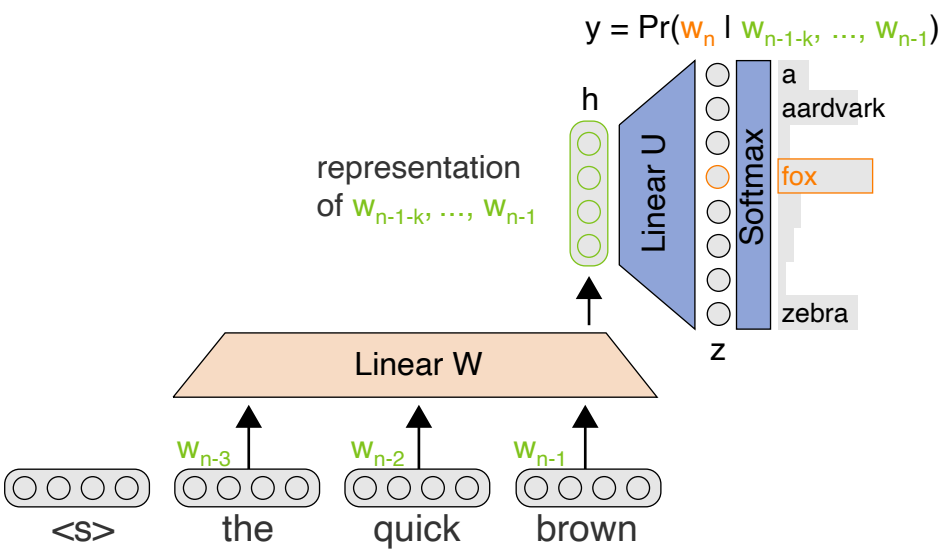
- Encode the history $w_1, \ldots, w_{n-1}$ into a latent representation.
- Predict w_n from this representation.



Language Modeling

Neural Language Models: Feed Forward

- The encoder can be a feed forward neural network.
- Due to the fixed input size, chose the latest k tokens for prediction.



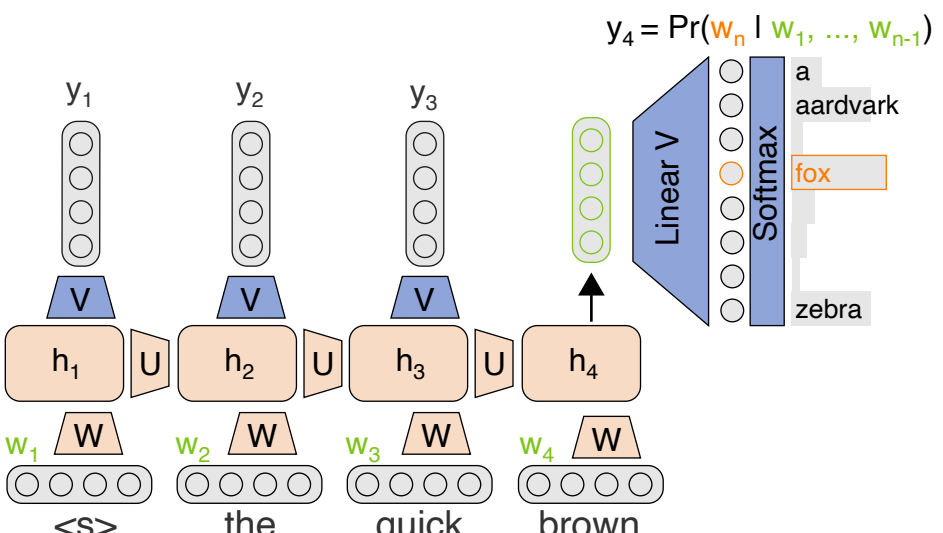
Forward computation:

$$\mathbf{e} = [\mathbf{w}_{n-1-k}; ...; \mathbf{w}_{n-1}]$$
$$\mathbf{h} = \sigma(\mathbf{We} + \mathbf{b})$$
$$\mathbf{y} = \text{softmax}(\mathbf{Uh})$$

Language Modeling

Neural Language Models: RNN

- The encoder can be a recurrent neural network.
- RNNs take sequences of arbitrary length as input and aggregate the history in a shared hidden layer h.
- For text generation, the output $\mathbf{y}_t$ can be used as input $\mathbf{w}_{t+1}$.



Forward computation:

$$\mathbf{h}_t = \sigma(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{w}_t)$$

$$\mathbf{y}_t = \text{softmax}(\mathbf{V}\mathbf{h}_t)$$

Remarks:

- RNN Notation: t incidates the timestep. $\mathbf{W}$ and $\mathbf{U}$ are the same at every t.
- Convolutional neural networks also work. However, many stacks are neccessary for long contexts and many stacks require residual and highway connections to train the CNN effectively.
- Due to the frequent multiplication of $\mathbf{h}$ and $\mathbf{U}$, RNNs face the *vanishing gradient problem* when training: the training signal becomes very small quickly and only signals at the end of the sequence have a meaningful impact.
- Long Short-Term Memory Networks (LSTMs) and Gated Recurrent Units (GRUs) can lessen the vanishing gradient by providing relevance, update, and forget gates in each cell. This greatly improves language modeling performance.

Language Modeling

Conditional Language Modeling

Definition 3 (Conditional Language Model)

A conditional language model is a probability distribution $P(w_1, ..., w_n|x)$ over all sequences of tokens $\langle w_1, ..., w_n \rangle \in V^*$ supported by a fixed-size vocabulary $\mathbf{V}$, given a source x as condition.

- In contrast, language models estimate the unconditional probability $P(w_1, ..., w_n)$
- The condition x can be a (distribution over) sequences but also other data like images, sound waves, or stock prices.

Common applications conditional language modeling are:

- Machine translation
- Summarization
- Paraphrasing
- Caption generation

Remarks:

- Conditional language models are the basis for large language models (LLMs).
- All architectures can be used for CLM. The usually best way are encoder-decoder models (originally from machine translation). The encoder ‘encodes’ the condition (prompt, sentence in source language, ...) and passes the encoding to the decoder as input. The decoder is then trained like an autoregressive language model, but with the additional encoding as input.