- In n-gram modeling, we take the conditional probability of the next word:

- And because of data sparsity must apply the Markov assumption, yielding:

- Neural language modeling is much more flexible and more powerful:
  - Unlike n-grams, NNs can directly model infinite context windows (eq 1) using one of two methods:
    - **Convolutional filtering**
    - **Recurrent network connections**
  - Like word2vec, unlike n-grams, learns a distributed representation your tokens
  - Unlike word2vec and n-grams which are linear models, NNs can model arbitrarily complex, highly non-linear relationships between tokens in a sentence!

Neural language modeling

11

$$P(\mathbf{x}^{(t)} \mid \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t-1)}) = P(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-n)}, \ldots, \mathbf{x}^{(t-1)}) \quad (2)$$

$$P(\mathbf{x}^{(t)} \mid \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t-1)}) \qquad (1)$$

We will discuss the practical limitations of these methods later in the class!

Neural language modeling

- In n-gram modeling, we take the conditional probability of the next word:

$$P(\mathbf{x}^{(t)} \mid \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t-1)}) \quad (1)$$

- And because of data sparsity must apply the Markov assumption, yielding:

$$P(\mathbf{x}^{(t)} \mid \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t-1)}) = P(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-n)}, \ldots, \mathbf{x}^{(t-1)}) \quad (2)$$

- Neural language modeling is much more flexible and more powerful:
  - Unlike n-grams, NNs can directly model infinite context windows (eq 1) using one of two methods:
    - **Convolutional filtering**
    - **Recurrent network connections**

    We will discuss the practical limitations of these methods later in the class!
  - Like word2vec, unlike n-grams, learns a distributed representation your tokens
  - Unlike word2vec and n-grams which are linear models, NNs can model arbitrarily complex, highly non-linear relationships between tokens in a sentence!

Neural language modeling: advantages

- Polysemy
  - Word2Vec and other static language models fail to capture polysemy. Having a single embedding per token limits our ability to capture the meaning of words. In fact, our training procedure, whereby identical words in different contexts are constrained to a single representation, introduces ambiguity into the model.

- Linearity
  - Word2Vec, and other similar static word embedding models such as GLOVE, are linear models. In contrast, neural networks are sequences of linear transformations separated by non-linearities; more layers layers translates to more representational power. Without non-linearities, no modeling capacity is gained by adding extra layers.

- Maximizing the context that a model can consider is an exercise that is largely computational in nature! The DL community has borrowed many of its most successful ideas from the field of discrete-time signal processing, and have built highly efficient numerical methods that utilize modern hardware (GPUs, TPUs).
  - CNNs
  - RNNs

- More recently (2017), a mechanism called ***attention*** has changed what is possible with language models. Transformer networks, based on this idea, **explicitly** represent contextual dependencies not only at the input layer but arbitrarily deep into the network. Transformers are the basis behind SOTA methods in NLP and vision! Methods based purely on attention come with the theoretical disadvantage of being able to only consider finite context windows; in practice we can make *n* large enough (10^3) for practical applications, and this context size ends up being larger than what is practical with convolutions and recurrent connections, anyway.