Natural Language Processing

Lecture 4
N-gram based language modeling

2021

Language models

What is a language model?

Recall that in formal language theory a language $\mathcal{L}$ is simply defined as a subset of Σ^* for some alphabet Σ.

Statistical language models, in contrast, switch to a *probabilistic view* of language production, and assign to any arbitrary $\langle w_1, \ldots, w_n \rangle \in V^*$ sequence of tokens from the vocabulary V a

$$P(\langle w_1, \ldots, w_n \rangle)$$

probability so that

$$\sum_{\mathbf{w} \in V^*} P(\mathbf{w}) = 1.$$

Vocabularies

Traditionally, the vocabulary of language models consisted of whole words, e.g.,

$$V = \{\textit{the}, \textit{be}, \textit{to}, \textit{of}, \ldots\}$$

but more recently subword and character based language models have also been widely used, with vocabularies like $\{$ *_don'*, *t*, *_un*, *related*, $\ldots\}$ or $\{$*a, b, c, d, e, f*, $\ldots\}$.

This lecture discusses word based language modeling techniques – techniques used for character and subword level modeling will be the subject of lectures 9 and 11.

Why are language models useful?

Probabilistic language models are important for a large number of NLP applications, in which the goal is to produce plausible word sequences as output, among them

- spell and grammar checking,
- predictive input,
- speech-to-text,
- chatbots,
- machine translation,
- summarisation.

Modeling with continuation probabilities

Using the chain rule, the probability of a token sequence $\mathbf{w} = \langle w_1, \dots, w_n \rangle$ can be rewritten as

$$P(\mathbf{w}) = P(w_1) \cdot P(w_2 | w_1) \cdots P(w_n | w_1, \dots, w_{n-1}),$$

that is, for a full language model it is enough to specify

(i) for any $w \in V$ word, the probability $P(w)$ that it will be the first word in a sequence, and

(ii) for any $w \in V$ and $\langle w_1, \dots, w_n \rangle$ partial sequence, the *continuation probability* for w, that is,

$$P(w \mid w_1, \dots, w_n).$$

Start and end symbols

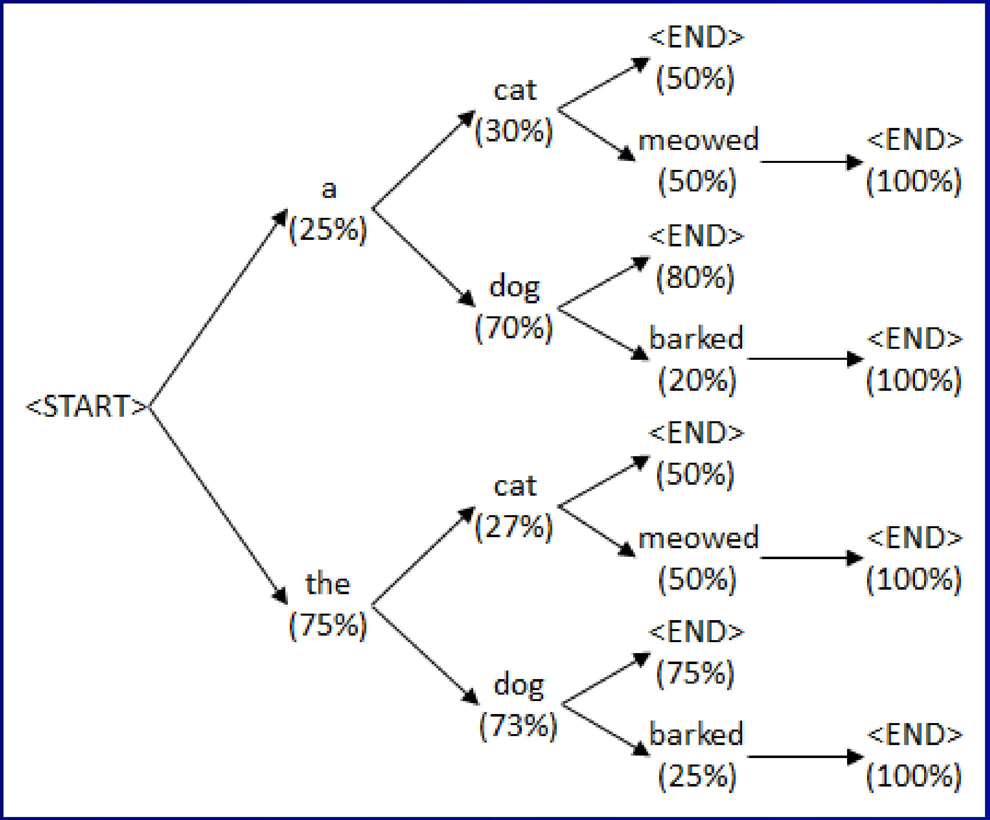
The chain rule based formulation of sequence probabilities

- requires a separate, unconditional clause for the starting probabilities, and
- does not address the probability of *ending* the sequence at a certain point.

Both issues can be solved by adding explicit $\langle\text{START}\rangle$ and $\langle\text{END}\rangle$ symbols to the vocabulary, and assuming that all sequences of the language start/end with these. With this trick the starting/ending probabilities can be rewritten in conditional form as $P(w \mid \langle\text{START}\rangle)$ and $P(\langle\text{END}\rangle \mid \mathbf{w})$.

Language model tree structure

Using start/end symbols the word sequences with their continuation probabilities assigned by an LM can be arranged in a tree structure:



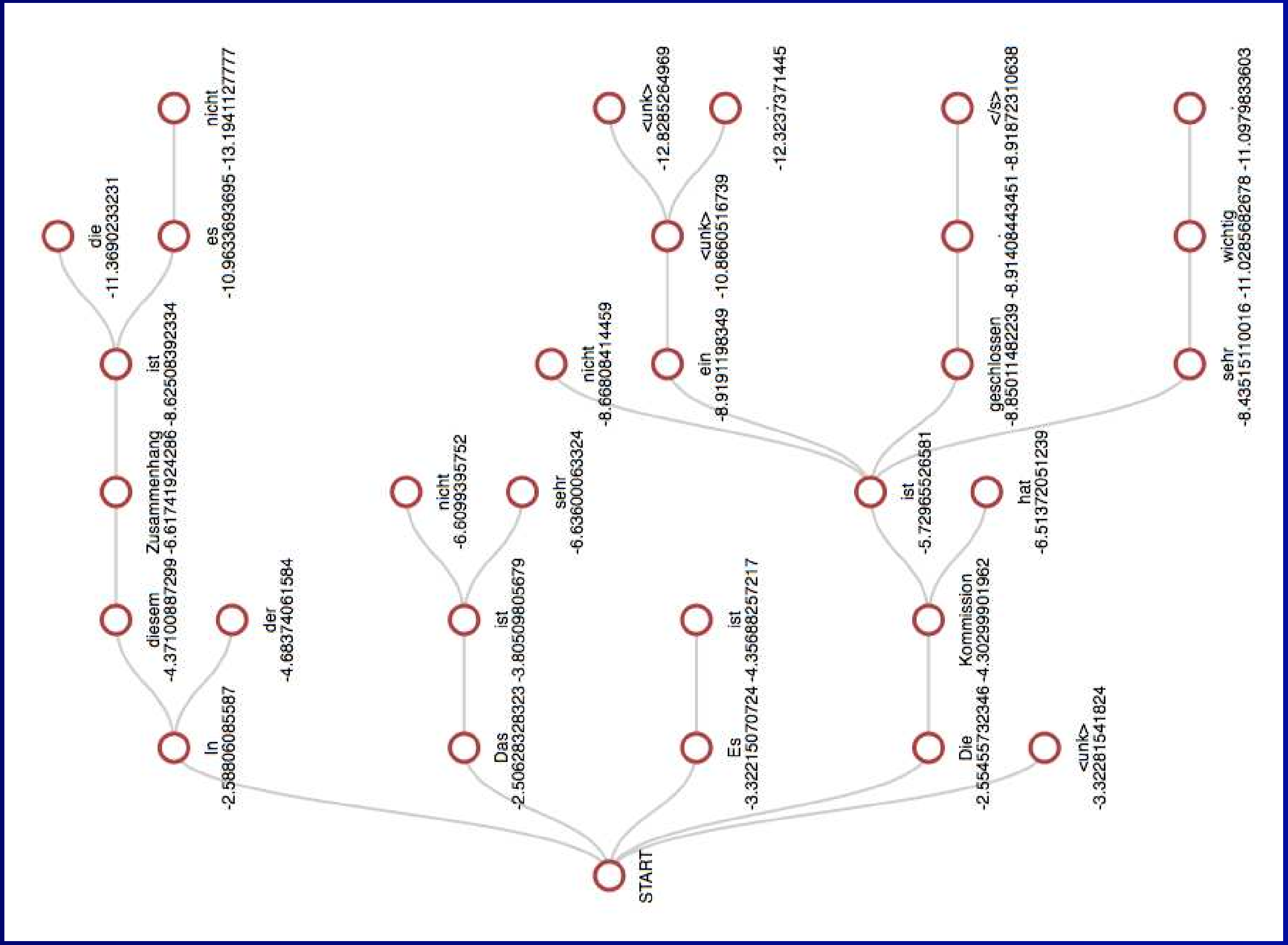
Text generation

Using a language model, new texts in the language can be generated on the basis of the model's generative probability distribution.
In terms of the tree structure shown on the previous slide, we are looking for branches on which the sum of weights (the log probabilities) are large. Exhaustive search is unfeasible, well-known strategies include

- greedy search,
- beam search, and
- stochaistic beam search.

Text generation cont.

A simple beam search example with $K = 5$:



Evaluation

Language model evaluation can be

- *extrinsic*: how well does the model do as a component in a spell checker, speech-to-text system etc., or
- *intrinsic*: how well the assigned probabilities correspond to the texts in a test corpus?

The most widely used intrinsic evaluation metric is *perplexity* on a corpus. A language model $\mathcal{M}$'s perplexity over the sequence $\mathbf{w} = \langle w_1, \ldots, w_n \rangle$ is

$$\mathbf{PP}_{\mathcal{M}}(\mathbf{w}) = \sqrt[n]{\frac{1}{P_{\mathcal{M}}(\mathbf{w})}}.$$

Evaluation cont.

With the chain rule perplexity can be rewritten as

$$\sqrt[n]{\frac{1}{P_{\mathcal{M}}(w_1)} \cdot \frac{1}{P_{\mathcal{M}}(w_2|w_1)} \cdots \cdot \frac{1}{P_{\mathcal{M}}(w_n|w_1,\ldots,w_{n-1})}}$$

which is exactly the *geometric mean* of the reciprocals of the conditional probabilities of all words in the corpus.

In other, words, perplexity measures, “how surprising”, on average, words (continuations) are in the corpus for the language model.

Evaluation cont.

Taking the logarithm of perplexity, with a few simple steps of algebraic manipulations we can see that the result is

$$-\frac{1}{n}\left(\log P_{\mathcal{M}}(w_1) + \sum_{i=2}^{n} \log P_{\mathcal{M}}(w_i \mid w_1, \ldots, w_{i-1})\right),$$

which is the average cross-entropy and negative log-likelihood per word. A simple consequence: by minimizing average cross-entropy or maximizing average log-likelihood one also minimizes the model's perplexitiy on the training data.

N-gram based modeling

Estimating probabilities

How can we estimate the required $P(\mathbf{w})$ probabilities from a corpus of texts? We could try to use occurrence counts to get the maximum likelihood estimate:

$$P(\mathbf{w}) \approx \frac{C(\mathbf{w})}{C(\text{all texts in corpus})}$$

but in any realistic corpus most texts occur only once and a lot of possible texts not at all. One option is switching to continuation probabilities:

$$P(w_i \mid w_1, \ldots, w_{i-1})$$

Estimating probabilities cont.

Using, again, count based estimation we could have

$$P(w_i \mid w_1, \ldots, w_{i-1}) \approx \frac{C(\langle w_1, \ldots, w_i \rangle)}{C(\langle w_1, \ldots, w_{i-1} \rangle)}$$

but with the same data sparsity problem. One way of alleviating it is to use the

$$P(w_i \mid w_1, \ldots, w_{i-1}) \approx P(w_i \mid w_{i-k}, \ldots, w_{i-1})$$

approximation for a certain k, using the assumption that the continuation probabilities are (approximately) determined by the previous last k tokens in the sequence.

N-gram language models

Using this approximation, the probability of a $\langle w_1, \ldots, w_n \rangle$ sequence can be calculated as

$$P(w_1) \prod_{i=2}^{k} P(w_i \mid w_1, \ldots, w_{i-1}) \prod_{i=k+1}^{n} P(w_i \mid w_{i-k}, \ldots, w_{i-1}),$$

and the big advantage is that the

$$P(w_i \mid w_{i-k}, \ldots, w_{i-1}) \approx \frac{C(\langle w_{i-k}, \ldots, w_i \rangle)}{C(\langle w_{i-k}, \ldots, w_{i-1} \rangle)}$$

estimates can be based only on the counts of maximum $k+1$ long subsequences in the corpus, so called N-grams ($N = 1, 2, 3, \ldots$).

Unigram models

The simplest N-gram language models are *unigram* models, assigning to a sequence $\langle w_1, \ldots, w_n \rangle$ the probability

$$P(w_1) \cdot P(w_2) \cdot \cdots \cdot P(w_{n-1}) \cdot P(w_n)$$

where the word probabilities can be estimated simply as

$$P(w) \approx \frac{C(w)}{\sum_{w' \in V} C(w')}.$$

Unigram models disregard the *order* of words and the most probable sequences are simply those entirely composed from the most frequent word(s).

Bigram models

Naturally, N-gram models based on longer subsequences are more fine-grained, even so called *bigram* models ($N = 2$) calculating sequence probabilities simply as

$$P(\langle w_1, \ldots, w_n \rangle) = P(w_1) \prod_{i=2}^{n} P(w_i \mid w_{i-1}),$$

with

$$P(w_2 \mid w_1) \approx \frac{C(\langle w_1, w_2 \rangle)}{C(w_1)}.$$

Markov language models

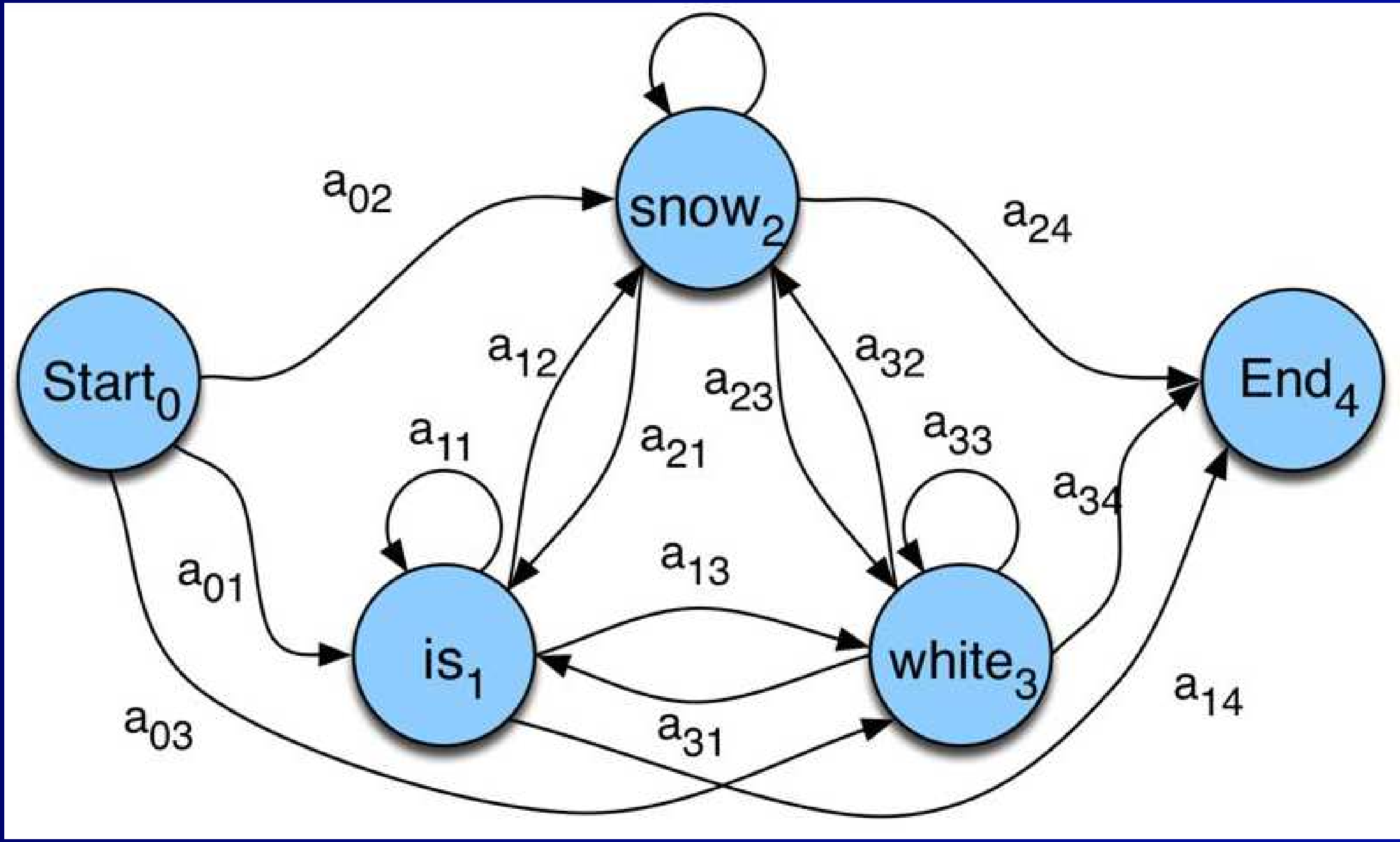
N-gram models, in effect, model language with probabilistic finite state machines (Markov models), in which the states correspond to $N-1$-grams.

E.g., in the case of an $\mathcal{M}$ bigram model, the states correspond to the vocabulary plus a start and end state, and the transition probabilities between states w_1 and w_2 are simply the $P(w_2 \mid w_1)$ continuation probabilities.

It is easy to see that the $P_{\mathcal{M}}(\mathbf{w})$ probability of a token sequence $\mathbf{w} = \langle w_1, \ldots, w_n \rangle$ is exactly the probability of the Markov model going through the states $\langle \text{START} \rangle, w_1, \ldots, w_n, \langle \text{END} \rangle$.

Markov language models cont.

A very simple Markov language model:



(Figure from D. Jurafsky's HMM slides)

Increasing N

Since in reality human languages are way too complex to satisfy a low-order Markov assumption, N-gram models with higher Ns (with $N = 3, 4$ or even 5) typically have better intrinsic and extrinsic performance. Unfortunately, the number of linguistically possible N-grams grows dramatically with N. E.g., in the 1,024,908,267,229 token N-gram corpus of Google the N-gram counts are:

- unigrams: 13,588,391
- bigrams: 314,843,401
- trigrams: 977,069,902
- fourgrams: 1,313,818,354
- fivegram: 1,176,470,663.

Increasing N cont.

The extremely high number of linguistically possible N-grams for higher N values poses two important problems:

- *data sparsity*: a lot of possible combinations will not occur even in large text corpora, or occur only very rarely, so it's difficult to estimate their probability;
- *model size*: even if estimates are correct, the model size will be enormous.

Smoothing

Additive smoothing

How can we solve the problem of N-grams that never or very rarely occur in the corpus? A simple solution is to *overcount* every N-gram by a certain number and use

$$P(w_i \mid w_{i-k}, \ldots, w_{i-1}) \approx \frac{C(\langle w_{i-k}, \ldots, w_i \rangle) + \delta}{C(\langle w_{i-k}, \ldots, w_{i-1} \rangle) + \delta|V|}.$$

The $|V|$ multiplier comes from the fact that for every $N-1$-gram there are exactly $|V|$ N-grams that are its continuations.

A widespread choice for δ is 1.

Additive smoothing cont.

An important problem with this solution:
If both $C(\langle w_1, w_2 \rangle) = 0$ and $C(\langle w_1, w_3 \rangle) = 0$, then under additive smoothing we have

$$p(w_1, w_2) = p(w_1, w_3).$$

Suppose now that w_2 is much more common than w_3.
Then, intuitively, we should have

$$p(w_1, w_2) > p(w_1, w_3)$$

instead of the above equality, so the result from additive smoothing seems wrong – we should somehow *interpolate* between unigram and bigram counts.

Interpolation

In case of bigrams, we add – with a certain weight – the probabilities coming from the unigram frequencies:

$$P(w_2 \mid w_1) \approx \lambda_1 \frac{C(\langle w_1, w_2 \rangle)}{C(w_1)} + (1 - \lambda_1) \frac{C(w_2)}{\sum_{w \in V} C(w)}$$

Recursive solution for arbitrary k:

$$P(w_{k+1} | w_1..w_k) \approx \lambda_k \frac{c(\langle w_1..w_{k+1} \rangle)}{c(\langle w_1..w_k \rangle)} + (1 - \lambda_k) P(w_{k+1} \mid w_2..w_k)$$

λ_k is empirically set on the basis of the corpus, typically using Expectation Maximization.