LIN7076 – Foundations of Computational Linguistic

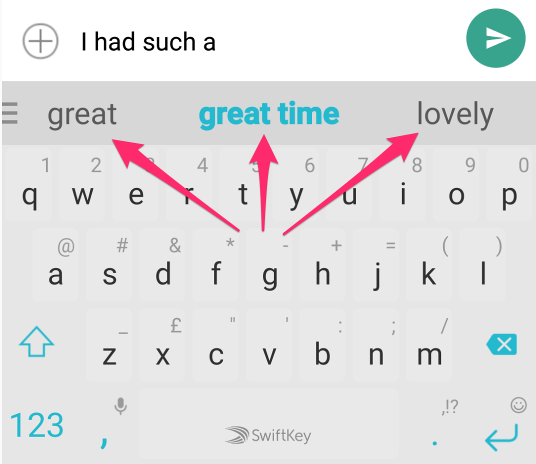
N-gram models

Adèle Hénot-Mortier

30/09/2025

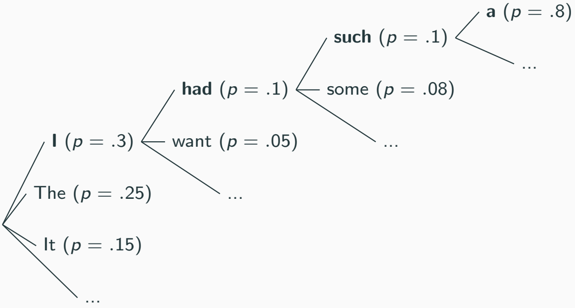
Queen Mary University of London

Generating text



The intuition behind n-grams

- You next word depends on the words ou have used before.
- When predicting a word, pick the most likely, **given wat was written before, within a specific window**.



- Likelihood (the ps) gets approximated by computing **statistics over a similar-enough training corpus**.

Plan for today

- Understand the *n*-gram **prediction process**.
- Understand how the probabilities used in that process are **approximated** from a training corpus.
- Understand how the resulting model can be **evaluated** against a test corpus.
- Explore **fixes/improvements** of shortcomings of the basic model.

The n-gram generation process

Why n-gram?

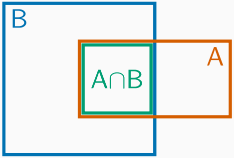
When predicting a word, pick the **most likely, given wat was written before** within a specific window.

- We want to compute the conditional probability of a word w_i, given a bunch of preceding words $[w_{i-n+1}, ..., w_{i-1}]$, that we call the context, of size $n - 1$.
- The sequence $[w_{i-n+1}, ..., w_{i-1}] + [w_i]$ is of size n. We call it a n-gram.
- For instance, the sequence $[I, had, such, a]$, is a 4-gram.
- A n-**gram model**, predicts the next word, given a $(n - 1)$-gram of preceding words.

Refresher on conditional probabilities

- The **conditional probability** of an event A, given an event B, can be computed using the joint probability of A and B, and B's probability.

$$P(A|B) = \frac{P(A, B)}{P(B)} = \frac{P(A \cap B)}{P(B)}$$



- Given any fixed event B, the probabilities $\{P(X|B) \mid X\}$, forms a probability distribution!

Rephrasing the n-gram intuition using conditional probabilities

- Our target even A, is that of the word w_i being used. We call this word target.
- Our given event B, is that of the words $[w_{i-n+1}, ..., w_{i-1}]$ being used before. We call this sequence context.
- We want to compute $P(A|B)$, the probability of wi given $[w_{i-n+1}, ..., w_{i-1}]$!

Rephrasing the n-gram intuition using conditional probabilities

- $P(w_i|[w_{i-n+1}, ..., w_{i-1}])$ depends solely on the context, and the context appended with the target (= the *n*-gram!).

$$
\begin{aligned}
P(A|B) &= \frac{P(A \cap B)}{P(B)} \\
P(w_i|[w_{i-n+1}, ..., w_{i-1}]) &= \frac{P([w_{i-n+1}, ..., w_i])}{P([w_{i-n+1}, ..., w_{i-1}])} \\
&= \frac{P(context + target)}{P(context)} \\
P(great|I\ had\ such\ a) &= \frac{P(I\ had\ such\ a\ great)}{P(I\ had\ such\ a)}
\end{aligned}
$$

- What would a **unigram model** look like?

A 3-gram example with made-up probabilities

- Suppose our sentence starts with the bigram *I had*. Suppose that *such* and *some* are the only 2 possible continuations.

$$
\begin{aligned}
P(\textit{such}|\textit{I had}) &= \frac{P(\textit{I had such})}{P(\textit{I had})} = .7 \\
P(\textit{some}|\textit{I had}) &= \frac{P(\textit{I had some})}{P(\textit{I had})} = .3
\end{aligned}
$$

- *Such* is more likely so we pick *such*. Now our sentence is *I had such*, and our context (bigram) becomes *had such*. Suppose that now *a* and *good* are the only 2 possible continuations.

$$
\begin{aligned}
P(\textit{a}|\textit{had such}) &= \frac{P(\textit{had such a})}{P(\textit{had such})} = .8 \\
P(\textit{good}|\textit{had such}) &= \frac{P(\textit{had such good})}{P(\textit{had such})} = .2
\end{aligned}
$$

- *A* is more likely, so we pick *a*. Now our sentence is *I had such a*, and our context becomes *such a*. Etc...

A potential issue with fixed window sizes

When predicting a word, pick the most likely, given wat was written before **within a specific window**.

- Why use a window, especially a short one? Natural language is recursive after all!
- Consider (1), which features **center-embedding**, in the form of an arbitrary number of nested complementizer phrases, between the main subject and its verb..[1]

(1) The $\underbrace{\textit{student}}_{\textit{subject}}$ who Jo said $\underbrace{\text{that Ed thinks that ... Al likes \_\_}}_{\text{arbitrarily long, nested relative clauses}}$ ...

- Could a *n*-gram model with a fixed window size correctly predict the number marking on the following verb? Could it even predict that it should be a verb?

[1] You can try to do the same with only relative clauses!

Why windows, anyway

- Using a fixed window makes the assumption that most of the time, language generation is oblivious of what was said too long ago. As we've just seen, this is not always true.
- This is known as the **Markov assumption**, which applies to processes well beyond CompLing! Biology, economy, robotics...
- Even if imperfect, the Markov assumption is computationally more efficient, as it require us to compute n-gram and $n-1$-gram **occurrences** for a fixed n, as we will soon see.
- It will also prevents the model from "memorizing" big chunks of the training data, something known as overfitting.
- Can you think of what shorter/longer windows would on average best capture in terms of syntax/semantics?

A note on sampling

When predicting a word, **pick the most likely**, given wat was written before within a specific window.

- What if two possible targets w_i and w'_i are associated to very close conditional probabilities, e.g. .999 and .1? Should we still pick the most likely, all the time?
- In practice, we **sample**: we pick target words randomly, giving more chances to the more likely ones.
- One simple way to think about it: drawing one ball from a big lottery box such that each target word is assigned a ball color, and gets as many balls as its conditional probability converted to a round percentage (so $p = .155$ means 16 balls).

Sampling example

- Remember our made up conditional probabilities for *such* and *some*, given the context *I had*?

$$P(such|I\ had) = \frac{P(I\ had\ such)}{P(I\ had)} = .7$$
$$P(some|I\ had) = \frac{P(I\ had\ some)}{P(I\ had)} = .3$$

- Instead of returning *such* (presumed most likely) 100% of the time, sampling will return *such* 70% of the time, and *some* 30% of the time.
- More generally, any target word w would be picked $(100 \times P(w|I\ had))\%$ of the time.

Approximating conditional probabilities

From probabilities to counts

- So far we have reasoned about hypothetical probabilities of words or sequences of words. **How are these probabilities estimated?**
- The probability of a sequence of words S of any fixed length (can be 1, i.e. S is a single word) can be estimated based on a training corpus.
- This is done by counting the number of S's occurrences, and dividing it up by the sum of the counts of all same-length sequences.
- The Google *n*-gram Viewer actually does this on a very large corpus of books!

From probabilities to counts

$$P(S) \quad = \quad \frac{C(S)}{\sum_{|S'|=|S|} C(S')} \triangleq \alpha C(S)$$

- Suppose now S' is obtained from S by removing S's last word: $S = S' + w$. If S is a n-gram, S' is a $(n-1)$-gram. S''s probability can be expressed by summing the probabilities of its immediate continuations...

$$P(S') \quad = \quad \sum_w P(S' + w) = \sum_w \alpha C(S' + w) = \alpha \sum_w C(S' + w) = \alpha C(S')$$

- We have just expressed probabilities of n and $(n-1)$-grams using their **counts** and the **same normalizing factor** α!

From conditional probabilities to counts

$$
\begin{aligned}
P([w_{i-n+1}, ..., w_i]) &= \alpha C([w_{i-n+1}, ..., w_i]) \\
P([w_{i-n+1}, ..., w_{i-1}]) &= \alpha C([w_{i-n+1}, ..., w_{i-1}])
\end{aligned}
$$

- Plugging this into our n-gram prediction formula, the αs eventually cancel out! So we get a conditional probability that **solely depends on n-gram counts**.

$$
\begin{aligned}
P(w_i|[w_{i-n+1}, ..., w_{i-1}]) &= \frac{P([w_{i-n+1}, ..., w_i])}{P([w_{i-n+1}, ..., w_{i-1}])} \\
&\simeq \frac{C([w_{i-n+1}, ..., w_i])}{C([w_{i-n+1}, ..., w_{i-1}])} \\
&\simeq \frac{C(\mathit{context} + \mathit{target})}{C(\mathit{context})}
\end{aligned}
$$

Dealing with sentence boundaries

$$P(w_i|[w_{i-n+1}, ..., w_{i-1}]) \quad \simeq \quad \frac{C([w_{i-n+1}, ..., w_i])}{C([w_{i-n+1}, ..., w_{i-1}])} = \frac{C(context + target)}{C(context)}$$

- The above equation predicts the next word, given a context of size $n-1$. But what should we do at the very beginning of a sentence, when the context is empty? And when should we stop?
- To properly initiate and terminate generation, we posit two extra silent "words": `<s>` ("**beginning** of sentence") and `</s>` ("**end** of sentence").
- For a n-gram model, each sentence in the training corpus gets **padded with** $n-1$ `<s>` **at the beginning, and** 1 `</s>` **at the end**. Counts incorporate these extra words.
- To start generating a sentence, we can then assume an initial, silent context made of $n-1$ `<s>` symbols!
- And we stop as soon as `</s>` gets generated.

Estimating conditional probabilities: a worked example

- Let's compute a toy bigram model. We pad with 1 <s> symbol.

(2) a. <s> my dog is nice </s>

b. <s> my cat is nasty </s>

c. <s> your cat hates my cat </s>

- Bigrams: (<s>, my), (my, dog), (dog, is), (is, nice), (nice, </s>), (<s>, my), (my, cat), (cat, is), (is, nasty), (nasty, </s>), (<s>, Your), (Your, cat), (cat, hates), (hates, my), (my, cat), (cat, </s>).
- Unigrams: <s>, my, dog, is, nice, </s>, <s>, my, cat, is, nasty, </s>, <s>, Your, cat, hates, my, cat, </s>.

Estimating conditional probabilities: a worked example

(3) a. <s> my dog is nice </s>

b. <s> my cat is nasty </s>

c. <s> your cat hates my cat </s>

$w_i \rightarrow$ $\downarrow w_{i-1}$	my	dog	is	nice	cat	nasty	your	hates	</s>	Total
<s>	2	0	0	0	0	0	1	0	0	3
my	0	1	0	0	2	0	0	0	0	3
dog	0	0	1	0	0	0	0	0	0	1
is	0	0	0	1	0	1	0	0	0	2
nice	0	0	0	0	0	0	0	0	1	1
cat	0	0	1	0	0	0	0	1	1	3
nasty	0	0	0	0	0	0	0	0	1	1
your	0	0	0	0	1	0	0	0	0	1
hates	1	0	0	0	0	0	0	0	0	1
Total	3	1	2	1	3	1	1	1	3	

$$P(my|\texttt{<s>}) = \frac{C(\texttt{<s>}\,my)}{C(\texttt{<s>})} = \frac{2}{3}$$

$w_i \rightarrow$ $\downarrow w_{i-1}$	my	dog	is	nice	cat	nasty	your	hates	</s>	Total
<s>	2	0	0	0	0	0	1	0	0	3
my	0	1	0	0	2	0	0	0	0	3
dog	0	0	1	0	0	0	0	0	0	1
is	0	0	0	1	0	1	0	0	0	2
nice	0	0	0	0	0	0	0	0	1	1
cat	0	0	1	0	0	0	0	1	1	3
nasty	0	0	0	0	0	0	0	0	1	1
your	0	0	0	0	1	0	0	0	0	1
hates	1	0	0	0	0	0	0	0	0	1
Total	3	1	2	1	3	1	1	1	3	

$$P(dog|my) = \frac{C(\text{my dog})}{C(my)} = \frac{1}{3}$$

$w_i \rightarrow$ $\downarrow w_{i-1}$	my	dog	is	nice	cat	nasty	your	hates	</s>	Total
<s>	2	0	0	0	0	0	1	0	0	3
my	0	1	0	0	2	0	0	0	0	3
dog	0	0	1	0	0	0	0	0	0	1
is	0	0	0	1	0	1	0	0	0	2
nice	0	0	0	0	0	0	0	0	1	1
cat	0	0	1	0	0	0	0	1	1	3
nasty	0	0	0	0	0	0	0	0	1	1
your	0	0	0	0	1	0	0	0	0	1
hates	1	0	0	0	0	0	0	0	0	1
Total	3	1	2	1	3	1	1	1	3	

$$P(\textit{hates}|\textit{cat}) = \frac{C(\text{cat hates})}{C(\textit{cat})} = \frac{1}{3}$$

$w_i \rightarrow$ $\downarrow w_{i-1}$	my	dog	is	nice	cat	nasty	your	hates	</s>	Total
<s>	2	0	0	0	0	0	1	0	0	3
my	0	1	0	0	2	0	0	0	0	3
dog	0	0	1	0	0	0	0	0	0	1
is	0	0	0	1	0	1	0	0	0	2
nice	0	0	0	0	0	0	0	0	1	1
cat	0	0	1	0	0	0	0	1	1	3
nasty	0	0	0	0	0	0	0	0	1	1
your	0	0	0	0	1	0	0	0	0	1
hates	1	0	0	0	0	0	0	0	0	-
Total	3	1	2	1	3	1	1	1	3	

- Can you guess which sentence(s) would be most likely to be generated by this bigram model?

Evaluating n-gram models

Probability of a corpus

- To evaluate how good a n-gram model is, one needs new data: typically a test corpus that is not too different from the training corpus. [2]
- We can then evaluate how likely this whole corpus is to be generated by our n-gram model!
- This is thanks to the chain rule:

$$\begin{aligned}
P(A_1 \cap A_2 \cap ... \cap A_k) &= P(A_1) \times \prod_{i=2}^{k} P(A_i | A_1 \cap ... \cap A_{i-1}) \\
&\overset{Markov}{=} P(A_1) \times \prod_{i=2}^{k} P(A_i | A_{i-n+1} \cap ... \cap A_{i-1}) \\
P([w_1 ... w_k]) &\overset{Markov}{=} P(w_1) \times \prod_{i=2}^{k} P(w_i | [w_{n-i+1} ... w_{i-1}])
\end{aligned}$$

[2] Why not test on the training corpus?

From probability to perplexity

- Measuring the performance of a n-gram based on the probability of a test corpus is nice, but depends on the corpus size: bigger corpora will mechanically be less probable (why?).
- To avoid this issue, we use a normalized variant of corpus probability called **perplexity**. The lower the perplexity, the better the model.

$$\begin{aligned} PPL([w_1...w_k]) &= P([w_1...w_k])^{-\frac{1}{k}} \\ &\overset{Markov}{=} \left(P(w_1) \times \prod_{i=2}^{k} P(w_i | [w_{n-i+1}...w_{i-1}]) \right)^{-\frac{1}{k}} \end{aligned}$$

- If $k \uparrow$, $\frac{1}{k} \downarrow$, $-\frac{1}{k} \uparrow$ and $p^{-\frac{1}{k}} \downarrow$ (with $p \geq 0$). So if two corpora have same probability, the larger one will get a smaller perplexity.

An issue with perplexity

$$PPL([w_1 ... w_k]) \overset{Markov}{=} \left(P(w_1) \times \prod_{i=2}^{k} P(w_i | [w_{n-i+1} ... w_{i-1}]) \right)^{-\frac{1}{k}}$$

- Looking at the above product of probabilities, we notice that if only one probability is 0, the whole product is $\frac{1}{0} = +\infty$!
- This would happen if the test corpus contains one n-gram that the training corpus does not have – which is far from unlikely!
- So perplexity is unlikely to distinguish good from bad n-gram models, as we defined them. Both kinds will likely get an infinite perplexity.
- More generally, probabilities learned from a finite training corpus will necessarily be 0 for many perfectly reasonable n-grams...bringing us to the competence-performance distinction.

A solution: smoothing

- To prevent perplexity from vanishing, we can simply add a small increment to all our *n*-gram counts!

$w_i \rightarrow$ $\downarrow w_{i-1}$	<s>	my	dog	is	nice	cat	nasty	your	hates	</s>	Total
<s>	ϵ	$2+\epsilon$	ϵ	ϵ	ϵ	ϵ	ϵ	$1+\epsilon$	ϵ	ϵ	$3+10\epsilon$
my	ϵ	ϵ	$1+\epsilon$	ϵ	ϵ	$2+\epsilon$	ϵ	ϵ	ϵ	ϵ	$3+10\epsilon$
dog	ϵ	ϵ	ϵ	$1+\epsilon$	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	$1+10\epsilon$
is	ϵ	ϵ	ϵ	ϵ	$1+\epsilon$	ϵ	$1+\epsilon$	ϵ	ϵ	ϵ	$2+10\epsilon$
nice	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	$1+\epsilon$	$1+10\epsilon$
cat	ϵ	ϵ	ϵ	$1+\epsilon$	ϵ	ϵ	ϵ	ϵ	$1+\epsilon$	$1+\epsilon$	$3+10\epsilon$
nasty	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	$1+\epsilon$	$1+10\epsilon$
your	ϵ	ϵ	ϵ	ϵ	ϵ	$1+\epsilon$	ϵ	ϵ	ϵ	ϵ	$1+10\epsilon$
hates	ϵ	$1+\epsilon$	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	$1+10\epsilon$
</s>	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	-
Total	-	$3+10\epsilon$	$1+10\epsilon$	$2+10\epsilon$	$1+10\epsilon$	$3+10\epsilon$	$1+10\epsilon$	$1+10\epsilon$	$1+10\epsilon$	$3+10\epsilon$	

- How do you think increasing/decreasing ϵ will affect training set perplexity? Test set perplexity?

How smoothing affects conditional probabilities

$$
\begin{aligned}
P(w_i|[w_{i-n+1}, ..., w_{i-1}]) &\simeq \frac{C_{smooth}([w_{i-n+1}, ..., w_i])}{C_{smooth}([w_{i-n+1}, ..., w_{i-1}])} \\
&\simeq \frac{C([w_{i-n+1}, ..., w_i]) + \epsilon}{C([w_{i-n+1}, ..., w_{i-1}]) + \epsilon|L|}
\end{aligned}
$$

- With $|L|$ the size of the lexicon (=set of unigrams).
- Why do we get $\epsilon|L|$ in the denominator and not just ϵ?
- Recall that the number of occurrences of a $(n-1)$-gram can be expressed as the number of occurrences of n-gram continuations.

$$
\begin{aligned}
C_{smooth}([w_{i-n+1}, ..., w_{i-1}]) &= \sum_{w \in L} C_{smooth}([w_{i-n+1}, ..., w_{i-1}, w]) \\
&= \sum_{w \in L} (C([w_{i-n+1}, ..., w_{i-1}, w]) + \epsilon) \\
&= \sum_{w \in L} C([w_{i-n+1}, ..., w_{i-1}, w]) + \epsilon|L| \\
&= C([w_{i-n+1}, ..., w_{i-1}]) + \epsilon|L|
\end{aligned}
$$

Another solution: stupid backoff

- Instead of messing with all the n-gram counts, one can revert to lower-order n-gram models, just in case the n-gram probability is 0.
- If for instance $P(\text{great}|\text{I had such a})$ is 0, then maybe $P(\text{great}|\text{had such a})$ is not? And if it is 0 too, then let's try $P(\text{great}|\text{such a})$! Etc.
- This kind of hybrid model can be expressed in the following, recursive[3] way (λ being some discounting factor).

$$SB(w_i|[w_{i-n+1}, ..., w_{i-1}]) = \begin{cases} P(w_i|[w_{i-n+1}, ..., w_{i-1}]) \text{ if } > 0 \\ \lambda SB(w_i|[w_{i-n+2}, ..., w_{i-1}]) \text{ otherwise} \end{cases}$$

- This model is no longer a proper probability distribution. Additionally, it requires us to compute k-gram occurrences for potentially every $k < n$!
- But unlike the smoothing solution, it may assign different probabilities to unlikely vs. super unlikely sequences.

[3] Notice how the "otherwise" case calls SB on a smaller context.