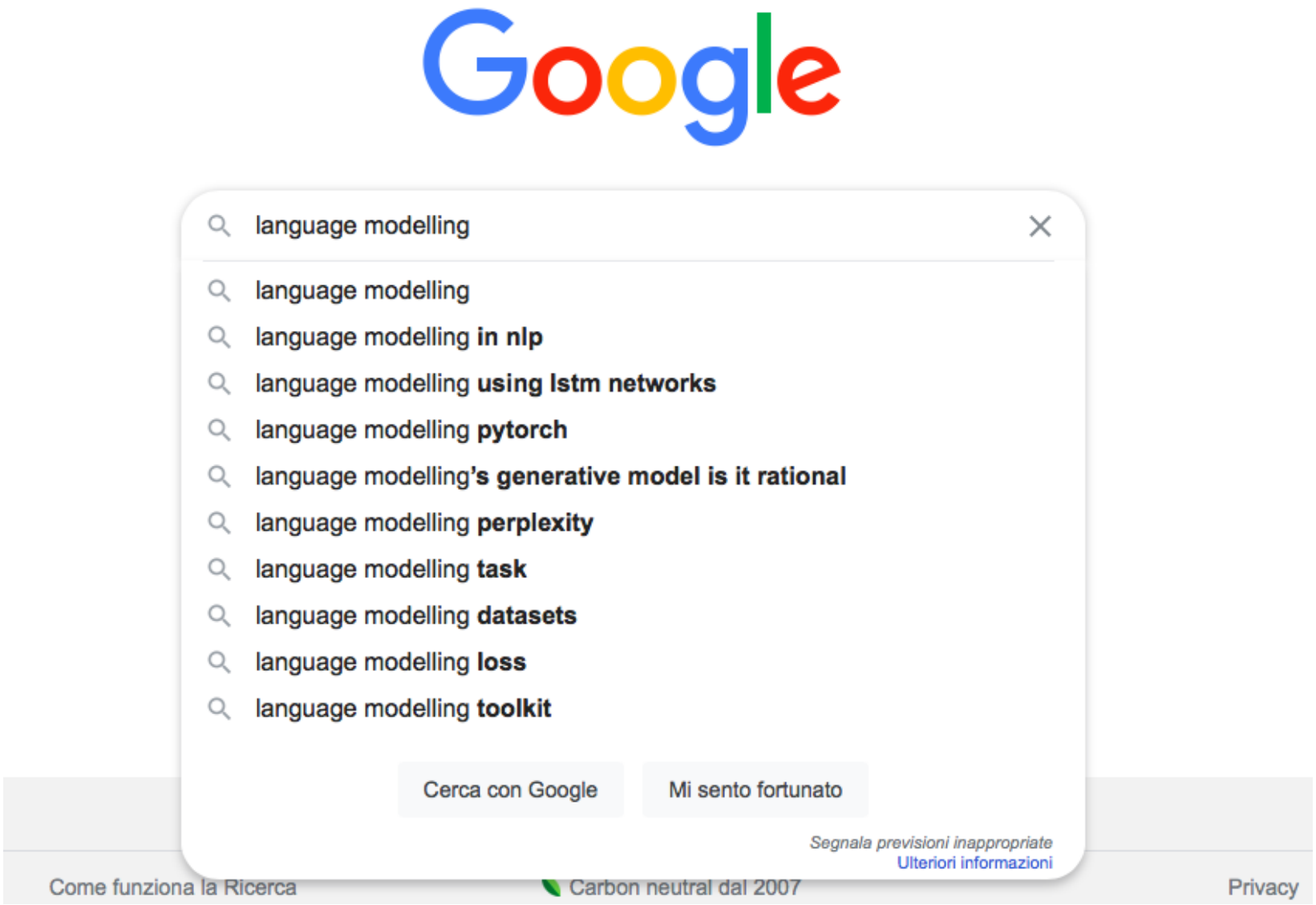
Natural Language Processing

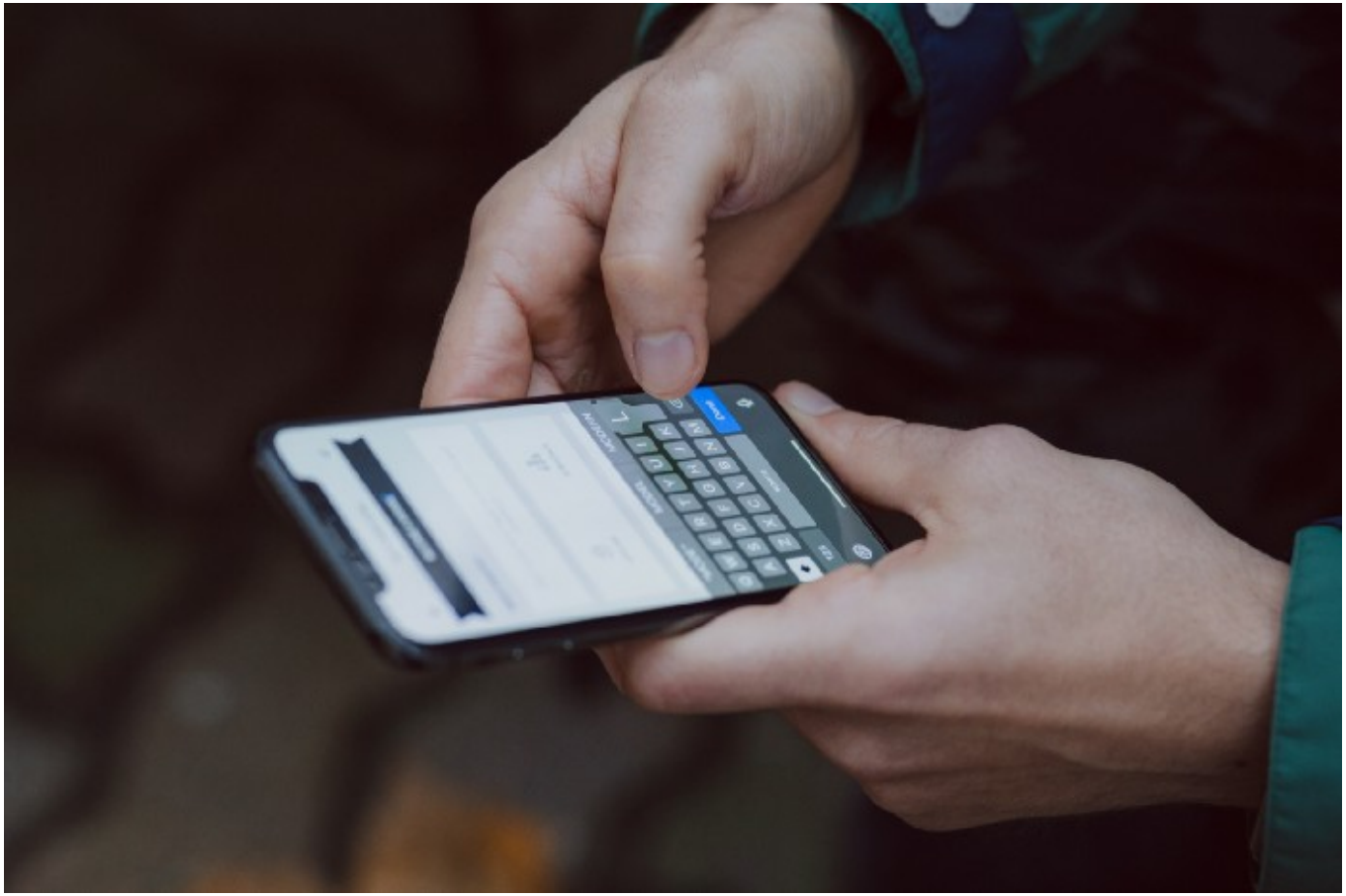
Lecture : Statistical Language Models

Master Degree in Computer Engineering
University of Padua
Lecturer : Giorgio Satta

Language modeling



Language modeling



Language modeling

Language modeling (LM) is the task of predicting which word comes next in a sequence of words. More formally, given a sequence of words $w_1 w_2 \cdots w_t$ we want to know the probability of the next word w_{t+1}:

$$P(w_{t+1} \mid w_1 w_2 \cdots w_t)$$

This allows language modeling to be treated as a classification task.

Notation : When string $w_1 w_2 \cdots w_t$ is understood from the context, we write $w_{1:t}$.

Language modeling

Rather than as **predictive** models, language models can also be viewed as **generative** models that assign probability to a piece of text:

$$P(w_1 \cdots w_t)$$

Not necessarily a complete sentence.

In this case, the model provides an answer to the question:

> *How likely is it that the given text belongs to the modeled language?*

Language modeling

These two views are **equivalent**, as the probability of a sequence can be expressed as a product of conditional probabilities:

$$P(w_{1:n}) = \prod_{t=1}^{n} P(w_t \mid w_{1:t-1})$$

This is the chain rule. For $t = 1$, we assume $P(w_t \mid w_{1:t-1}) = P(w_1)$.

Conversely, a conditional probability can be expressed as a ratio of two sequence probabilities:

$$P(w_{t+1} \mid w_{1:t}) = \frac{P(w_{1:t+1})}{P(w_{1:t})}$$

We have applied here the definition of conditional probaility.

Applications

In a machine translation system, we are given the following Chinese sentence

他	向	记者	介绍了	主要	内容	
he	*to*	*reporters*	*introduced*	*main*	*content*	(lit.)

We get three candidate translations:

- he introduced reporters to the main contents of the statement
- he briefed to reporters the main contents of the statement
- he briefed reporters on the main contents of the statement

We use a language model to select the most likely candidate.

Same idea for speech recognition and spelling correction.

Probability estimation

©indiamart

Probability estimation

Assume the text '*its water is so transparent that*'. We want to know the probability that the next word is '*the*'.

Let $C(\textit{its water is so transparent that})$ be the number of times the string is seen in a large corpus.

One way to estimate the above probability is to set

$$P(\textit{the} \mid \textit{its water is so transparent that}) = \frac{C(\textit{its water is so transparent that the})}{C(\textit{its water is so transparent that})}$$

To make these probabilities sum up to one, we need to add to the vocabulary V a sentence end marker. In this way, every token in the corpus is *always* followed by some symbol.

Probability estimation

In general, given a large corpus, we can set:

$$P(w_t \mid w_{1:t-1}) \quad = \quad \frac{C(w_{1:t})}{C(w_{1:t-1})}$$

Under previous assumptions, we have $C(w_{1:t-1}) = \sum_u C(w_{1:t-1}u)$, where u ranges over all symbols (tokens and sentence end marker).

Quantities $C(w_{1:t})$ are called **frequencies**.

The ratio $C(w_{1:t})/C(w_{1:t-1})$ is called **relative frequency**.

The estimator above is therefore called the **relative frequency estimator**.

Probability estimation

The relative frequency estimator will be correct in the limit, that is, with infinite data.

In practice, consider an aggressive upper bound of $N = 20$ words in our sequences, and an English vocabulary V of size $|V| = 10^5$. Then the number of possible sequences is $|V|^{20} = 10^{100}$.

This estimator is extremely data-hungry, and suffers from **high variance**: depending on what data happens to be in the corpus, we could get very different probability estimations.

We need to introduce some bias into the model.

N-gram model

©Sunny Srinidhi

N-gram model

A string $w_{t-N+1:t}$ of N words is called ***N*-gram**.

The ***N*-gram model** approximates the probability of a word given the entire sentence history by conditioning only on the past $N-1$ words.

The assumption that the probability of a word depends only on a few previous words is called a Markov assumption.

The 2-gram model, for example, makes the approximation

$$P(w_t \mid w_{1:t-1}) \approx P(w_t \mid w_{t-1})$$

1-gram model is just the single word probability $P(w_t)$, unconditioned.

N-gram model

The general equation for the N-gram model is

$$P(w_t \mid w_{1:t-1}) \approx P(w_t \mid w_{t-N+1:t-1})$$

The relative frequency estimator for the N-gram model is then

$$P(w_t \mid w_{t-N+1:t-1}) = \frac{C(w_{t-N+1:t})}{C(w_{t-N+1:t-1})}$$

N-gram model

For $N = 2$ we have

$$\begin{aligned} P(w_t \mid w_{t-1}) &= \frac{C(w_{t-1}w_t)}{\sum_u C(w_{t-1}u)} \\ &= \frac{C(w_{t-1}w_t)}{C(w_{t-1})} \end{aligned}$$

For $N = 1$ we have $P(w_t) = \frac{C(w_t)}{n}$ where n is the length of the training set.

Alternatively, we can view $n = \sum_u C(u)$ as the number of 1-gram observations, where u ranges over all tokens excluding the sentence end marker.

The model requires estimating and storing the probability of only $|V|^N$ events, which is exponential in N (a constant), not in the length of the sentence.

Example

Consider a mini-corpus of three sentences. We augment each sentence with start and end markers <s> and </s>:

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

We get the following 2-gram probabilities, among others:

$$P(\texttt{I}|\texttt{<s>}) = \frac{2}{3} = .67 \qquad P(\texttt{Sam}|\texttt{<s>}) = \frac{1}{3} = .33$$
$$P(\texttt{am}|\texttt{I}) = \frac{2}{3} = .67 \qquad P(\texttt{</s>}|\texttt{Sam}) = \frac{1}{2} = .50$$
$$P(\texttt{Sam}|\texttt{am}) = \frac{1}{2} = .50 \qquad P(\texttt{do}|\texttt{I}) = \frac{1}{3} = .33$$

Example

Let us look into a more realistic corpus, from a dialogue system that answers questions about restaurants.

Bigram counts from a sample of the sentences in the corpus:

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Majority of the values are zero, despite words have been sampled to cohere with each other.

Example

Unigram counts:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Bigram probabilities after normalization:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Example

Here are a few other useful probabilities:

$$P(\texttt{I}|\texttt{<s>}) = 0.25 \qquad P(\texttt{english}|\texttt{want}) = 0.0011$$
$$P(\texttt{food}|\texttt{english}) = .5 \qquad P(\texttt{</s>}|\texttt{food}) = .68$$

We can now compute the probability of sentence 'I want English food':

$$\begin{aligned} & P(\texttt{</s> i want english food </s>}) \\ = \; & P(\texttt{i}|\texttt{<s>}) \times P(\texttt{want}|\texttt{i}) \times P(\texttt{english}|\texttt{want}) \\ & \times P(\texttt{food}|\texttt{english}) \times P(\texttt{</s>}|\texttt{food}) \\ = \; & 0.25 \times 0.33 \times 0.0011 \times 0.5 \times 0.68 \\ = \; & 0.000031 \end{aligned}$$

Example

What kinds of linguistic phenomena are captured in these bigram statistics?

Syntactic aspects

- after verb **eat** we usually see a noun or an adjective
- after infinitive marker **to** we usually have a verb

Other aspects

- high probability of sentences beginning with pronoun **I** (task)
- higher probability that people are looking for Chinese versus English food (cultural)

N is a hyperparameter of the model. When setting its value, we face the **bias-variance tradeoff**.

When N is too small (high bias), it fails to recover long-distance word relations, as for instance in:

> *The* **computer** *that is on the 3rd floor of our office building* **crashed**.

When N is too large, we get data sparsity (high variance).

Learning

The relative frequency estimator can be mathematically derived by maximizing the **likelihood** of the dataset.

This can be done by solving a **constrained optimization problem**, using Lagrange multipliers.

More precisely, we maximize the log of the product of all sentence probabilities, under the constrain that probabilities sum up to one.

Therefore the relative frequency estimator is also called the **maximum likelihood estimator** (MLE).

Practical issues

To compute 3-gram probabilities for words at the start of a sentence, we use two **markers**, e.g. $P(\texttt{I}|\texttt{<s><s>})$. Similarly for higher order N-grams.

Multiplying many small probabilities results in **underflow**. It is much safer and more efficient to use **negative log probabilities**, $-\log(p)$, and add them.

Note the minus sign: the smaller the probability, the larger the $-\log$.

For web-scale datasets (large vocabulary), the model has **huge space** requirements, even for small values of N.

Evaluation

Evaluation

Extrinsic evaluation : Use the model in some application (e.g. speech recognition) and measure performance of that application.

Difficult to do reliably, time consuming.

Intrinsic evaluation : Look at performance of model in isolation, with respect to a given evaluation measure.

Perplexity, see next slide.

Perplexity

Intrinsic evaluation of language models is based on the inverse probability of the test set, normalized by the number of words.

For a test set $W = w_1 w_2 \cdots w_n$ we define **perplexity** as:

$$\begin{aligned} \mathrm{PP}(W) &= P(w_{1:n})^{-\frac{1}{n}} \\ &= \sqrt[n]{\prod_{j=1}^{n} \frac{1}{P(w_j \mid w_{1:j-1})}} \end{aligned}$$

We need to discuss inverse probabilities and n-th root of product.

Perplexity

The multiplicative inverse probability $1/P(w_j \mid w_{1:j-1})$ can be seen as a measure of how surprising the next word is.

Equivalently, how much information is carried out by the next word.

The degree of the root averages over all words of the test set, providing **average surprise per word**.

The perplexity of two language models is only comparable if they use identical vocabularies.

For large enough test data obtained in a uniform way, perplexity is more or less constant, i.e. independent of n.

The lower the perplexity, the better the model.

Example

Evaluation of *N*-gram models on the *Wall Street Journal*.

Training set: 38 million tokens;
Vocabulary: 19,979 types;
Test set: 1.5 million words.

	Unigram	Bigram	Trigram
Perplexity	962	170	109

Practical issues

Make sure to use a training corpus that has a similar **genre** to whatever task you are trying to accomplish.

An (intrinsic) improvement in perplexity **does not** guarantee an (extrinsic) improvement in the performance of a language processing task like speech recognition or machine translation.

Sampling sentences

Random sampling for LM means that we iteratively choose the next word to generate according to its probability given the context.

Thus we are more likely to generate words that have a high probability and less likely to generate words that have a low probability.

Sampling sentences

We write $x \sim p(x)$ to mean 'choose x by sampling from the distribution $p(x)$.

Let $w_1, w_2, \ldots, w_N$ be the sequence of words to be generated.

$$
\begin{aligned}
&\mathrm{i} \leftarrow 1 \\
&w_i \sim \mathrm{p}(w) \\
&\textbf{while } w_i \text{ != EOS} \\
&\quad \mathrm{i} \leftarrow \mathrm{i} + 1 \\
&\quad w_i \sim \mathrm{p}(w_i \mid w_{<i})
\end{aligned}
$$

Sampling sentences

Random sentences generated from 1-gram, 2-gram, 3-gram, and 4-gram models trained on Shakespeare's works.

Corpus: 884,647 tokens, 29,066 types. Sentence generation samples the probabilities of the model.

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; –It cannot be but so.

The 3-gram and 4-gram sentences look a lot like Shakespeare. Indeed, for many 4-grams, there is only one continuation.

Sparse data

$$P(w_t \mid w_{t-N+1:t-1}) = \frac{C(w_{t-N+1:t})}{C(w_{t-N+1:t-1})}$$

If there isn't enough data in the training set, counts will be zero for some grammatical sequences.

Then some of the N-gram probabilities will be **zero** or **undefined**.

Perplexity on the test set will also be undefined.

There are three scenarios we need to consider:

- zero numerator: smoothing
- zero denominator: backoff, interpolation
- out-of-vocabulary words in test set: estimation of unknown words

Smoothing

Susan Wilkinson on Unsplash

Smoothing

Smoothing techniques (also called **discounting**) deal with words that are in our vocabulary V but were never seen before in the given context (zero numerator).

Smoothing prevents LM from assigning **zero probability** to these events.

Idea :

- shave off a bit of probability mass from more frequent events
- give it to the events we have never seen in the training set

Laplace smoothing

Laplace smoothing does not perform well enough, but provides a useful baseline.

Idea : Pretend that everything occurs once more than the actual count.

In this way, all the counts that used to be 0 will now have a count of 1, the counts of 1 will be 2, and so on.

Also known as add-one smoothing.

Laplace smoothing

In order to apply smoothing to our N-gram model, let us rewrite the relative frequency estimator in a more convenient form:

$$\begin{aligned} P(w_t \mid w_{t-N+1:t-1}) &= \frac{C(w_{t-N+1:t})}{C(w_{t-N+1:t-1})} \\ &= \frac{C(w_{t-N+1:t})}{\sum_u C(w_{t-N+1:t-1}u)} \end{aligned}$$

In the summation, u is a single word ranging over the entire vocabulary V and the sentence end marker.

Recall that every word token is followed by some symbol, either a word or the end marker.

Laplace for 1-grams

Let n be the number of tokens, that is, the length of the training set, and recall that $|V|$ is the number of word types.

We assume the vocabulary V is fixed.

Recall that the 1-gram model relative frequency estimator is:

$$P(w_t) = \frac{C(w_t)}{n}$$

The **adjusted estimate** of the probability of word $w_t \in V$ is then:

$$P_{\mathrm{L}}(w_t) = \frac{C(w_t) + 1}{n + |V|}$$

The extra $|V|$ comes from pretending there are $|V|$ more observations, one for each word type.

Laplace for 1-grams

Alternatively, we can think of P_{L} as applying an **adjusted count** C^* to the n actual observations:

$$
\begin{aligned}
C^*(w_t) &= (C(w_t)+1)\frac{n}{n+|V|} \\
P_{\mathrm{L}}(w_t) &= \frac{C^*(w_t)}{n} \\
&= \frac{C(w_t)+1}{n+|V|}
\end{aligned}
$$

Under this view, the smoothing algorithm amounts to **discounting** (lowering) counts for high frequency words and redistributing:

$$\sum_{u \in V} C(u) = \sum_{u \in V} C^*(u) = n$$

Laplace for 2-grams

The 2-gram model relative frequency estimator is

$$\begin{aligned} P(w_t \mid w_{t-1}) &= \frac{C(w_{t-1}w_t)}{\sum_u C(w_{t-1}u)} \\ &= \frac{C(w_{t-1}w_t)}{C(w_{t-1})} \end{aligned}$$

The adjusted estimate of the probability of 2-gram $w_{t-1}w_t$ is then:

$$\begin{aligned} P_{\mathrm{L}}(w_t \mid w_{t-1}) &= \frac{C(w_{t-1}w_t)+1}{\sum_u [C(w_{t-1}u)+1]} \\ &= \frac{C(w_{t-1}w_t)+1}{C(w_{t-1})+|V|} \end{aligned}$$

Laplace for 2-grams

The adjusted count is:

$$C^*(w_t \mid w_{t-1}) = \frac{[C(w_{t-1}w_t)+1]C(w_{t-1})}{C(w_{t-1})+|V|}$$

$P_{\mathrm{L}}(w_t \mid w_{t-1})$ is larger than $P(w_t \mid w_{t-1})$ for 2-gram sequences that occur zero or few times in the training set.

However, $P_{\mathrm{L}}(w_t \mid w_{t-1})$ will be much lower (too low) for 2-gram sequences that occur often. So Laplace smoothing is **too crude** in practice.

Add-k smoothing

Add-k smoothing is a generalization of add-one smoothing.

Also known as Lidstone smoothing

For some $0 \leqslant k < 1$:

$$P_{\text{Add-k}}(w_t \mid w_{t-1}) = \frac{C(w_{t-1}w_t) + k}{C(w_{t-1}) + k|V|}$$

Jeffreys-Perks law corresponds to the case $k = 0.5$, which works well in practice and benefits from some theoretical justification.

See for instance (Manning and Schutze, 1999).

Smoothing and perplexity

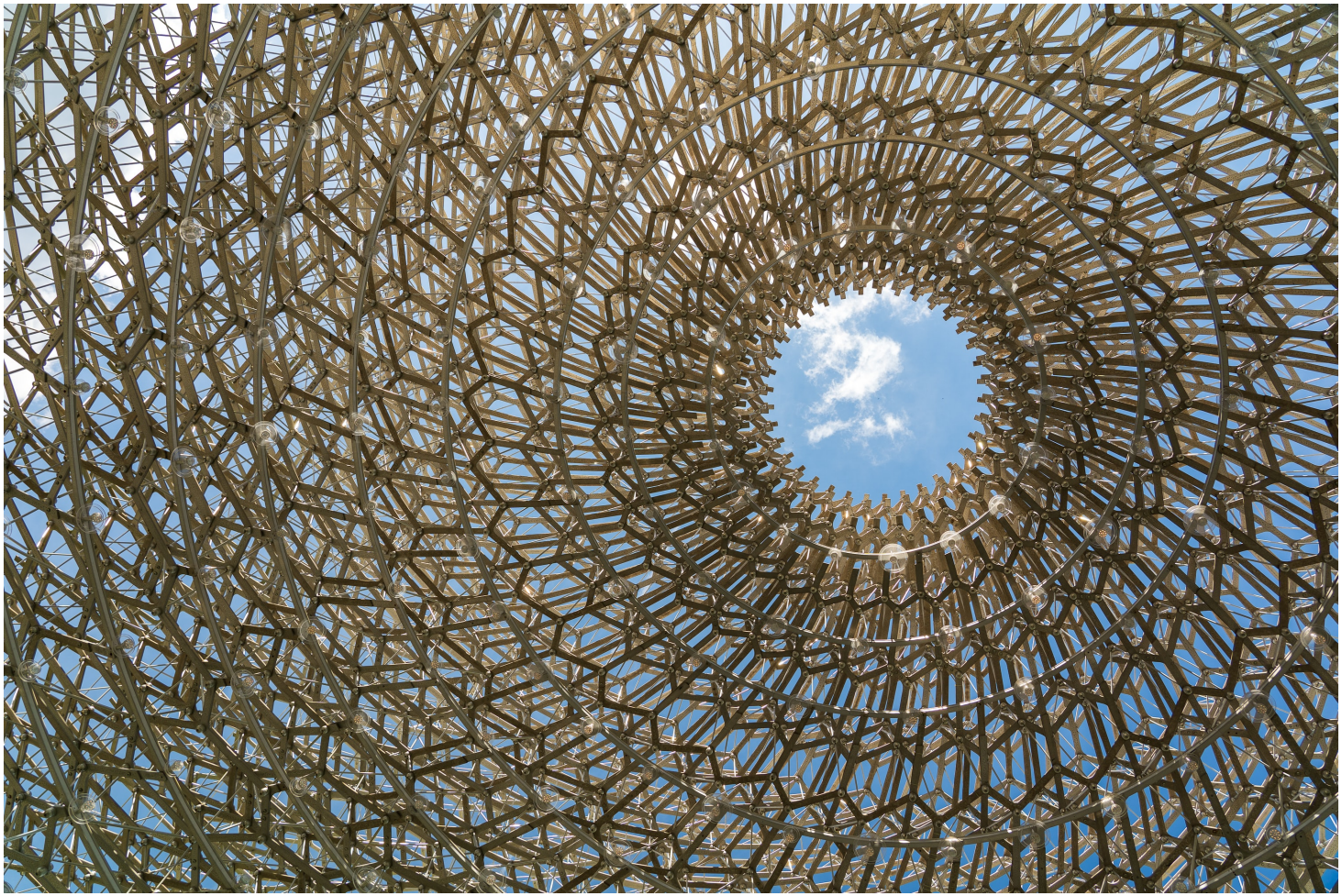
When smoothing a language model, we are redistributing probability mass to outcomes we have never observed.

This leaves a smaller fraction of the probability mass to the outcomes that we actually did observe during training.

The training data becomes less likely.

Thus, the more probability we are taking away from observed outcomes, the **higher** the perplexity on the training data.

Backoff and interpolation



Nick Fewings on Unsplash

Backoff and interpolation

Backoff and interpolation techniques deal with words that are in our vocabulary, but in the test set combine to form previously unseen contexts.

These techniques prevent LM from creating **undefined probabilities** for these events (zero-divide).

Backoff

Backoff combines fine grained models (large N) with coarse grained models (low N).

Idea :

- if you have trigrams, use trigrams
- if you don't have trigrams, use bigrams
- if you don't even have bigrams, use unigrams

Katz backoff is a popular but rather complex algorithm for backoff.

Katz backoff is often combined with a smoothing method called Good-Turing.

Stupid backoff

With very large text collections (web-scale) a rough approximation of Katz backoff is often sufficient, called **stupid backoff**.

For some small λ:

$$P_S(w_t \mid w_{t-N+1:t-1}) = \begin{cases} P(w_t \mid w_{t-N+1:t-1}) = \frac{C(w_{t-N+1:t})}{C(w_{t-N+1:t-1})}, \\ \qquad \text{if } C(w_{t-N+1:t}) > 0 \\ \lambda P_S(w_t \mid w_{t-N+2:t-1}), \\ \qquad \text{otherwise} \end{cases}$$

It is not difficult to show that P_S is **not a probability distribution**. However, in practical settings stupid backoff turns out to be effective.

Linear interpolation

In **simple linear interpolation**, we combine different order N-grams by linearly interpolating all the models.

Simple linear interpolation for $N = 3$

$$\begin{aligned} P_L(w_t \mid w_{t-2}w_{t-1}) &= \lambda_1 P(w_t \mid w_{t-2}w_{t-1}) + \\ &\quad \lambda_2 P(w_t \mid w_{t-1}) + \\ &\quad \lambda_3 P(w_t) \end{aligned}$$

for some choices of positive λ_1, λ_2, λ_3 such that $\sum_j \lambda_j = 1$

Upper-order models set to zero when undefined. Note that we use lower-order models even in cases we have non-zero counts for the upper-order ones.

Linear interpolation

What are good choices for the λ_j's? Algorithms exist that attempt to optimise likelihood of training data.

We might have different values for the λ_j's, depending on sequences $w_{t-2}w_{t-1}$, subject to

$$\sum_j \lambda_j(w_{t-2}w_{t-1}) = 1$$

The more frequent $w_{t-2}w_{t-1}$, the more reliable the trigram probabilities, the higher we can choose $\lambda_1(w_{t-2}w_{t-1})$.

Unknown Words

Annie Spratt on Unsplash

Unknown Words

Unknown words, also called **out of vocabulary** (OOV) words, are words we haven't seen before.

Replace by new word token <UNK> all words that occur fewer than d times in the training set, d some small number.

Proceed to train LM as before, treating <UNK> as a regular word.

At test time, replace all unknown words by <UNK> and run the model.

Limitations

N-gram language models have several **limitations**.

Scaling to larger N-gram sizes is problematic, both for computational reasons and because of increased sparsity.

Smoothing techniques are intricate and require careful engineering to retain a well-defined probabilistic interpretation.

Without additional effort, N-gram models are unable to share statistical strength across similar words.

Observations of 'red apple' do not affect estimates for 'green apple'.