Introduction: What is Language Modelling?

Language modelling is the task of predicting what word comes next in a text sequence.

Language Models in NLP:

- Humans have some "probability" of natural language
- But what is the probability of a random sequence of words? -> Hard to compute
  ==> we define a LM to estimate the probability of text sequences, this LM is **good** if it will give a higher probability to a correct sequence

  Utility examples:
  predicting next word or emoji in chat, auto-correction, search engine recommendations, machine translation , Speech recognition, Summarization, Question answering...

Probability:

Joint Probability

- **P(A and B):** the probability of event A and B occurring together
- **P(W = "I saw a cat on a mat"):** $P(\text{I, saw, a, cat, on, a, mat})$

Conditional Probability

- **P(A|B):** The probability that event A occurs, given that event B occurs.
- $P(\text{mat} \mid \text{I saw a cat on a}) = P(\text{mat} \mid \text{I, saw, a, cat, on, a})$

Probability of a Word

What is an event in language? A linguistic unit (text, sentence, word, token, symbol).
$V = \{\text{the, be, to, of, ...}\}$: a fixed vocabulary

What is likely a word in a sentence $w \in V$? Compute the probability over the vocabulary
$P(w = \text{"the"}) = \frac{\text{occurrences of } w}{\text{number of words in text}}$

Probability of a Sentence

Goal: Estimate the probability of a sentence $P(W = w_1, w_2, \ldots, w_T)$ occurring in a language (English).

We don't have true probabilities of all languages in the world but instead, we have data (a large collection of books, Internet...). We will use this data to **estimate** the probability of W even if the word sequence is not in the corpus!

Estimate Sentence Probability

Can we apply the same approach we use for words on sentences?
->No because we already said that we need to estimate the probability of W even if it is not in the corpus , for example :
$P(\text{themut is tinming the tebn}) = \frac{0}{|\text{corpus}|} = 0$
$P(\text{mut the tinming tebn is the}) = \frac{0}{|\text{corpus}|} = 0$
but in this case the first sentence is more likely than the second and that's why we never treat sentences like **atomic units** like words

Sentence probability

We estimate sentence probabilities by combining probabilities of smaller parts that occur more frequently like word or phrases.
--> the intuition behind N-gram language models
The chain rule : $P(\text{A and B}) = P(A|B)P(B) = P(B|A)P(A)$
Factorize the sentence probability: left-to-right :
$P(W = w_1, w_2, \ldots, w_T) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2)\ldots P(w_T|w_1w_{2\ldots}w_{T-1})$
the general formula is :
$P(W = w_1, w_2, \ldots, w_T) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2)\ldots P(w_T|w_1w_{2\ldots}w_{T-1}) = \prod_i^T P(w_i|$

N-gram language models

LMs can be simply modelized by the equation we introduced earlier :
$P(W = w_1, w_2, \ldots, w_t) = \prod_i^T P(w_i|w_1 \ldots w_{i-1}) = \prod_i^T P(w_i|w_{<i})$
What is the difference between one LM and the other in this case? it's in how the conditional probability $P(w_i|w_{<i})$ or $P(w_i|w_1 \ldots w_{i-1})$ is computed.

- the first computation method is statistical from the text corpora:
  $P(w_T|w_1w_2 \ldots w_{T-1} = \frac{N(w_1\ldots w_T)}{N(w_1\ldots w_{T-1})}$
  but many of these are unseen in the data
  exp : to get $P(\text{every villain is a hero in his own mind })$ we still need $P(mind|\text{Every villain is a hero is his own})$ -> count can be 0

N-grams

Markov assumption:

The probability of a word only depends on a fixed number of previous words. For an **n-gram** model: $P(w_T|w_1w_{2\ldots}w_{t-1})$≈$P(w_T|w_{T-n+1}\ldots w_{T-1})$

Definition

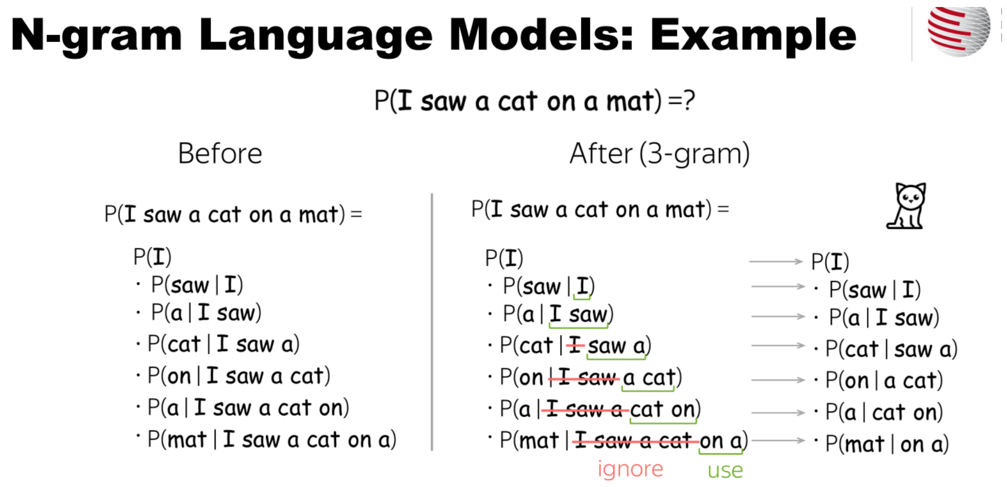
An n-gram is a chunk of n consecutive words
Unigram:hello , photosynthesis
Bigram: blue sky, dark sky

Trigram: cats chase mice, mice chase cats

4-gram: cats always chase mice

N-gram language Models

Unigram:$P(w_T|w_1w_2...w_{t-1})≈P(w_T)$
Bigram: $P(w_T|w_1w_2...w_{t-1})≈P(w_T|w_{T-1})$
Trigram:$P(w_T|w_1w_2...w_{t-1})≈P(w_T|w_{T-2}w_{T-1})$



N-gram Language Models: Example

P(I saw a cat on a mat) =?

Before

P(I saw a cat on a mat) =

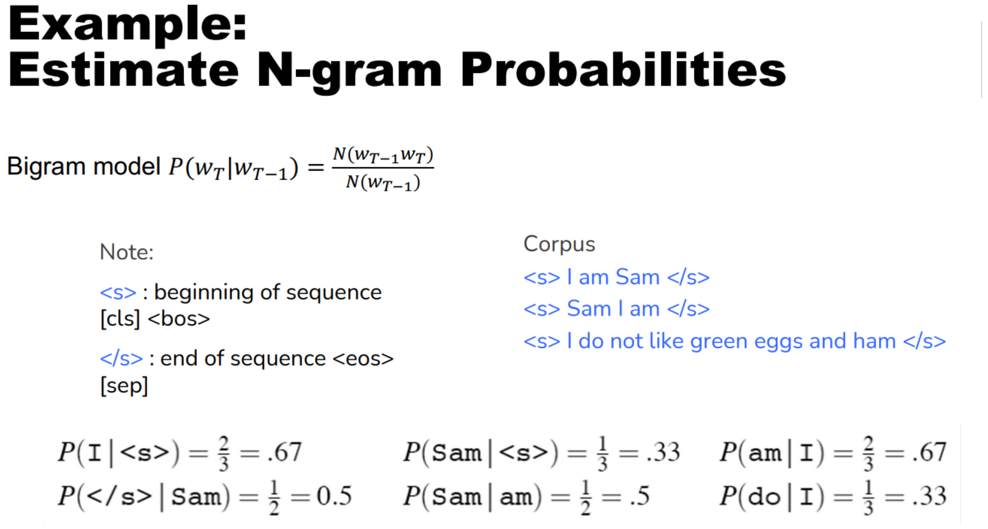
P(I)
· P(saw | I)
· P(a | I saw)
· P(cat | I saw a)
· P(on | I saw a cat)
· P(a | I saw a cat on)
· P(mat | I saw a cat on a)

After (3-gram)

P(I saw a cat on a mat) =

P(I) → P(I)
· P(saw | I) → · P(saw | I)
· P(a | I saw) → · P(a | I saw)
· P(cat | ~~I~~ saw a) → · P(cat | saw a)
· P(on | ~~I saw~~ a cat) → · P(on | a cat)
· P(a | ~~I saw a~~ cat on) → · P(a | cat on)
· P(mat | ~~I saw a cat~~ on a) → · P(mat | on a)

ignore use



Example: Estimate N-gram Probabilities

Bigram model $P(w_T|w_{T-1}) = \frac{N(w_{T-1}w_T)}{N(w_{T-1})}$

Note:

<s> : beginning of sequence [cls] <bos>

</s> : end of sequence <eos> [sep]

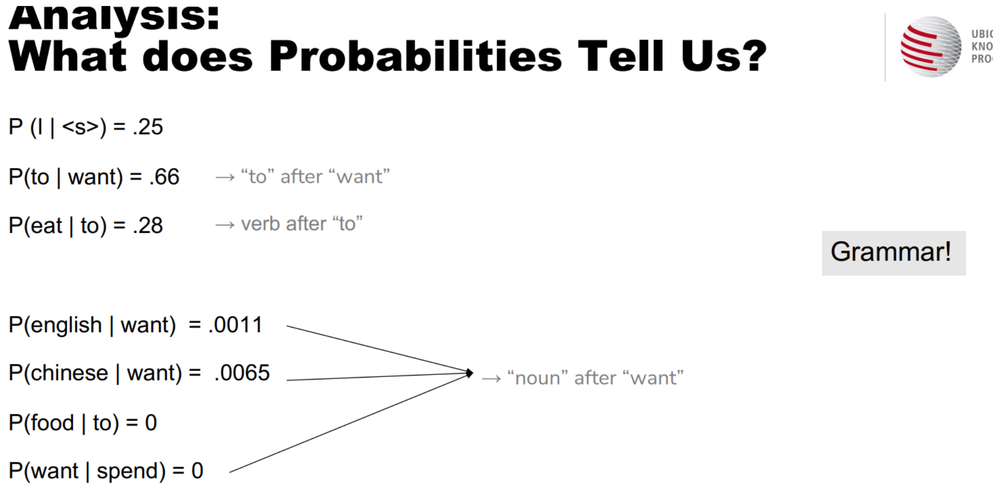
Corpus
<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

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

$P(\texttt{</s>}\,|\,\texttt{Sam}) = \frac{1}{2} = 0.5$ $P(\texttt{Sam}\,|\,\texttt{am}) = \frac{1}{2} = .5$ $P(\texttt{do}\,|\,\texttt{I}) = \frac{1}{3} = .33$

Negative Log-probability Models

Word probabilities are usually very small -> multiplying them gets so tiny we can't represent the probability accurately even with double-precision floats
Solution: calculate in log space
why? because log maps the probability values that are between 0 and 1 to negative log values between 0 and -∞
we add negative log values instead of multiplying(great advantage since adding is faster)
Unigram: $log(\prod_i P(w_i)) = \sum_i log(P(w_i))$
Bigram: $log(\prod_i P(w_i|w_{i-1})) = \sum_i log(P(w_i|w_{i-1}))$



Analysis: What does Probabilities Tell Us?

P (I | <s>) = .25

P(to | want) = .66 → “to” after “want”

P(eat | to) = .28 → verb after “to”

Grammar!

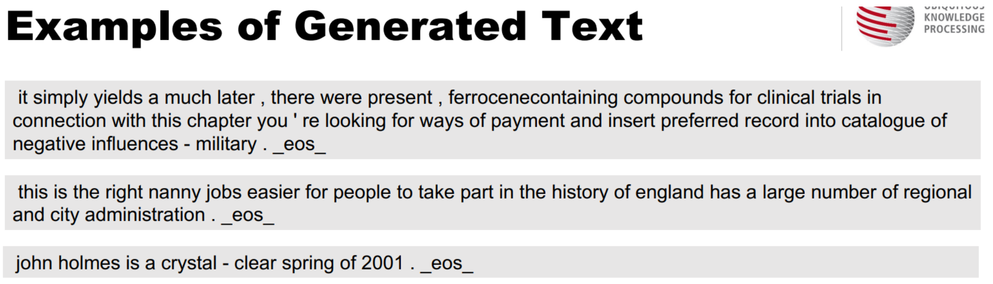
P(english | want) = .0011

P(chinese | want) = .0065

P(food | to) = 0

P(want | spend) = 0

→ “noun” after “want”



Examples of Generated Text

it simply yields a much later , there were present , ferrocenecontaining compounds for clinical trials in connection with this chapter you ' re looking for ways of payment and insert preferred record into catalogue of negative influences - military . _eos_

this is the right nanny jobs easier for people to take part in the history of england has a large number of regional and city administration . _eos_

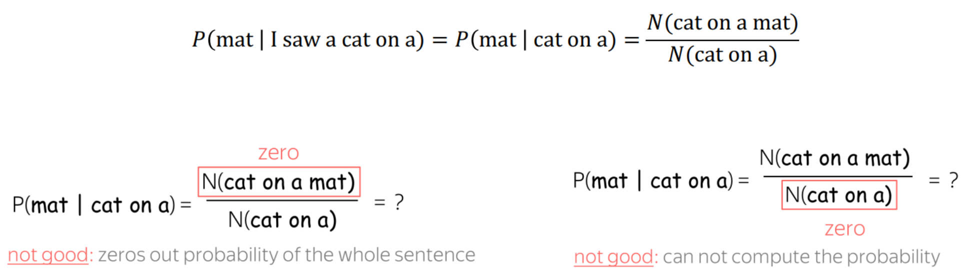
john holmes is a crystal - clear spring of 2001 . _eos_

What is clearly wrong with these samples?

These samples are not fluentthe model does not use long context, and relies only on a few words.

The **inability to use long contexts** is the main shortcoming of n-gram models.

Sparsity problems with N-grams



$$P(\text{mat} \mid \text{I saw a cat on a}) = P(\text{mat} \mid \text{cat on a}) = \frac{N(\text{cat on a mat})}{N(\text{cat on a})}$$

$$P(\text{mat} \mid \text{cat on a}) = \frac{\overset{\text{zero}}{\boxed{N(\text{cat on a mat})}}}{N(\text{cat on a})} = ?$$

not good: zeros out probability of the whole sentence

$$P(\text{mat} \mid \text{cat on a}) = \frac{N(\text{cat on a mat})}{\underset{\text{zero}}{\boxed{N(\text{cat on a})}}} = ?$$

not good: can not compute the probability

Solution :Laplace Smoothing (assume we see all probabilities at east one time add 1 to all counts or alternatively a small α)

$$P(\text{mat}|\text{cat on a}) = \frac{\alpha + N(\text{cat on a mat})}{\alpha \cdot |V| + N(\text{cat on a})}$$

Back-off Method

$N(\text{cat on a}) = 0$ -> try "on a"

$P(mat|\text{cat on a}) \approx P(mat|\text{on a})$

$N(\text{on a}) = 0$ try "a"

$P(mat|\text{on a}) \approx P(mat|\text{a})$

$N(\text{a}) = 0$ try **Unigram**

$P(mat|\mathrm{a}) \approx P(mat)$
In summary use less context for ones we don't know much about
if no trigram try bigram if not try unigram and so on

Evaluating LMs

How good is our model?

Does our model prefer good sentences to bad one?(assign higher probability to correct sentences and lower it for grammatically incorrect or rarely seen ones)

How to measure this?

Extrinsic Evaluation:measure performance on a downstream application:
put the model on a downstream task like spelling corrector or speech recognizer.
But this is time consuming and we still need an evaluation measure for the downstream task
Intrinsic Evaluation :design a measure inherent to the current task.
Can be much quicker and easier because we directly measure the model on the current task
But not always easy to figure out the right measure: ideally, one that correlates well with extrinsic
measures.
Lets define an intrinsic measure for LMs: Perplexity
The Best LM is one that predicts unseen texts (higher likelihood and less perplexity)
it gives the highest $P(W)$

- Perplexity is the inverse probability of the unseen texts, normalized by the number of words.

$$\text{Perplexity}(w_1 w_2 \dots w_N) = 2^{-\frac{1}{N} P(w_1 w_2 \dots w_N)}$$

Perplexity

- the best perplexity is 1:If the model is perfect and assigns prob 1 to correct tokens, then the log-probability = 0, and the perplexity = 1
- the worst perplexity in $|V|$:If the model knows nothing about the data, it thinks every word have the same prob 3 |7| regardless of context

$$\text{Perplexity}(w_1 w_2 \dots w_N) = 2^{-\frac{1}{N} P(w_1 w_2 \dots w_N)} = 2^{-\frac{1}{N} \sum_i^N log_2 P(w_i | w_1 \dots w_{i-1})} = 2^{-\frac{1}{N} N log_2 \frac{1}{|V|}} = 2^{log_2 |V|} = |V|$$

Perplexity is always a value between 1 and $|V|$