# Natural Language Understanding, Generation, and Machine Translation (2021–22)

*School of Informatics, University of Edinburgh*
*Alexandra Birch*

## Tutorial 1: Language Models (Week 2)

Questions on this worksheet that ask for an expression or calculation generally have one correct answer, and are designed to concretely explore aspect of these models just beyond what we discussed in lecture, by asking you about some implications of model design. These questions are not intended to be difficult, but notice that they are not primarily about recalling information—they require you to engage with the material and pay attention to the details. You should expect that some of the *easier* exam question may be of this form.

We also include some more open-ended questions that you will need to think about. It is intended to help you see how modeling choices impact the number of parameters, and to think about strategies for reducing this number. We will discuss some strategies in Tutorial 2.

## 1 N-gram Models

Typically, we use techniques like smoothing or backoff in conjunction with $n$-gram models.

**Question 1:** Smoothing

a. What is a problem that these techniques help to mitigate?

b. Why does this problem arise in classic $n$-gram models?

**Solution 1:** Smoothing

a. Smoothing and backoff are used to ensure that no $n$-gram probabilities are zero. They also may help improve estimates for observed but rare $n$-grams.
b. Maximum likelihood estimation of a classic $n$-gram model results in zero probability for any unobserved $n$-gram.

Let's consider how different modeling assumptions affect the number of parameters in a language model. Suppose you have a trillion tokens of web text to train a language model. The text contains:

- 2.6 billion unique unigrams,
- 15 billion unique bigrams,
- 62 billion unique trigrams,
- 157 billion unique 4-grams, and
- 264 billion unique 5-grams

These are type-level statistics, so this implies that there are 2.6 billion word types in the corpus.

Now suppose that you want to estimate from this data a **classic $n$-gram model** in which each $n$-gram $w_{i-n+1}...w_i$ has an associated parameter for the conditional probability $P(w_i \mid w_{i-n+1}, ..., w_{i-1})$.

**Question 2:** Parameter Counts

a. *In theory*, the number of parameters in an $n$-gram model is equivalent to the number of possible $n$-grams given a vocabulary $V$. That is, every possible $n$-gram has an associated parameter, whether or not it is observed in the data. How would you express this theoretical number of parameters in terms of the order $n$ and a vocabulary $V$ with $|V|$ unique words?

b. Given the data above, calculate the theoretical number of parameters for $n$ from 1 through 5.
2.6; (2.6)^2; (2.6)^3;(2.6)^4; (2.6)^5

c. *In practice*, most $n$-gram models only store non-zero parameters. Given the data above, what fraction of the theoretical parameters are non-zero, for $n$ from 1 through 5?
2.6/2.6; 15/(2.6)^2

**Solution 2:** Parameter Counts

The $n$-gram statistics are real. They originate from
`https://kheafield.com/papers/stanford/crawl_paper.pdf`

a. $|V|^n$

b.
- $2.6 \times 10^9$ unigrams
- $6.7 \times 10^{18}$ bigrams
- $1.76 \times 10^{28}$ trigrams
- $4.67 \times 10^{37}$ 4-grams
- $1.19 \times 10^{47}$ 5-grams

c. Notice that $n$-grams at this scale are *incredibly* sparse!
- 1
- $2 \times 10^{-9}$ bigrams
- $3.5 \times 10^{-18}$ trigrams
- $3.36 \times 10^{-27}$ 4-grams
- $2.21 \times 10^{-36}$ 5-grams

From these numbers, it should be clear that in practice, we should only store non-zero parameters, and that we should use smoothing or backoff. So for a given $n$, we need to store not only parameters for order $n$, but for all lower orders as well.

**Question 3:** Implementation of Parameter Counts

a. Ignoring parameters of the smoothing or backoff method (e.g. interpolation parameters), how many parameters do we need to store for a 5-gram model learned from our data?
2.6+15+62+157+264

b. If each parameter requires 4 bytes of memory, what is the minimum amount of memory needed to store the model? (Note: This is a *very* optimistic lower bound on the size of the model. We need to index the $n$-grams in order to look them up, and most index data structures take substantial space!)
*4

**Solution 3:** Implementation of Parameter Counts

a. 500 billion ($5 \times 10^{11}$)

b. Approximately 2 terabytes. The paper linked above will show why this is an optimistic lower bound—a *very* compact and efficient implementation required 5.6 terabytes.