# Introduction to NLP (CS7.401)

Spring 2022, IIIT Hyderabad
11 Jan, Tuesday (Lecture 3)

Taught by Prof. Manish Shrivastava

## Evaluation Metrics

Perplexity and entropy are information theoretic metrics which help in measuring how well a language model models a corpus.

Entropy describes how much information there is about what the next word will be. For example, for a bigram model, let $x$ range over bigrams with the probability function $p(x)$; then the entropy of the model is

$$H(X) = -\sum_{x=1}^{n} p(x) \log p(x).$$

It is the lower bound on the number of bits it takes to encode information about bigram likelihood.

Cross entropy is an upper bound on entropy derived from estimating true entropy by a subset of possible strings (when we don't know the real probability distribution).

Perplexity measures how "confused" the model is – it calculates the weighted average branching factor, or the average number of choices that can be made over all choice points, weighted by their probabilities of occurrence.

## Smoothing

When calculating the probabilities of bigrams, we don't know whether the zeroes are in fact non-occurring or simply low-frequency. This is dealt with by a process called smoothing.

Smoothing is done to account for the part of the language to which the model has possibly not been exposed. There are many ways to carry out smoothing.

## Laplace Smoothing

This is the oldest smoothing method available. Each unseen $n$-gram is given a very low estimate.

Let $N$ be the number of seen $n$-grams and $B$ the number of possible $n$-grams ($V^2$ in the case of bigrams). Then we add 1 to the counts of *all* possible bigrams, which changes the probabilities to

$$\frac{n+1}{N+B}.$$

This is a simple method to avoid zeroes, but causes problems. For example, it causes almost 50% of all the probability mass goes to unknowns.

## Good-Turing Smoothing

The basic principle in this algorithm is to estimate the zero-count $n$-grams by using the counts of single-count $n$-grams (called singletons). Thus the Good-Turing algorithm uses the frequency of singletons as a re-estimate of the frequency of zero-count $n$-grams.

We assume that things that have a similar frequency have similar properties, or behave in a similar way (syntactically). Thus we can apply this assumption to zero-count $n$-grams and singletons – words that appear once can be considered, statistically, practically the same as ones that never appear.

Let $N_c$ be the number of $n$-grams with count $c$. The Good-Turing estimate replaces the frequency of all $N_c$ words having frequency $c$ with a smoothed count, given by

$$c^* = (c+1)\frac{N_{c+1}}{N_c},$$

for every $c$. For example, if $N_2 = 100, N_3 = 90$, then for $c = 2$,

$$c^* = 3 \cdot \frac{90}{100} = 2.7,$$

which is the updated count.
This gives us the updated probability of things with zero counts as

$$\frac{N_1}{N}.$$