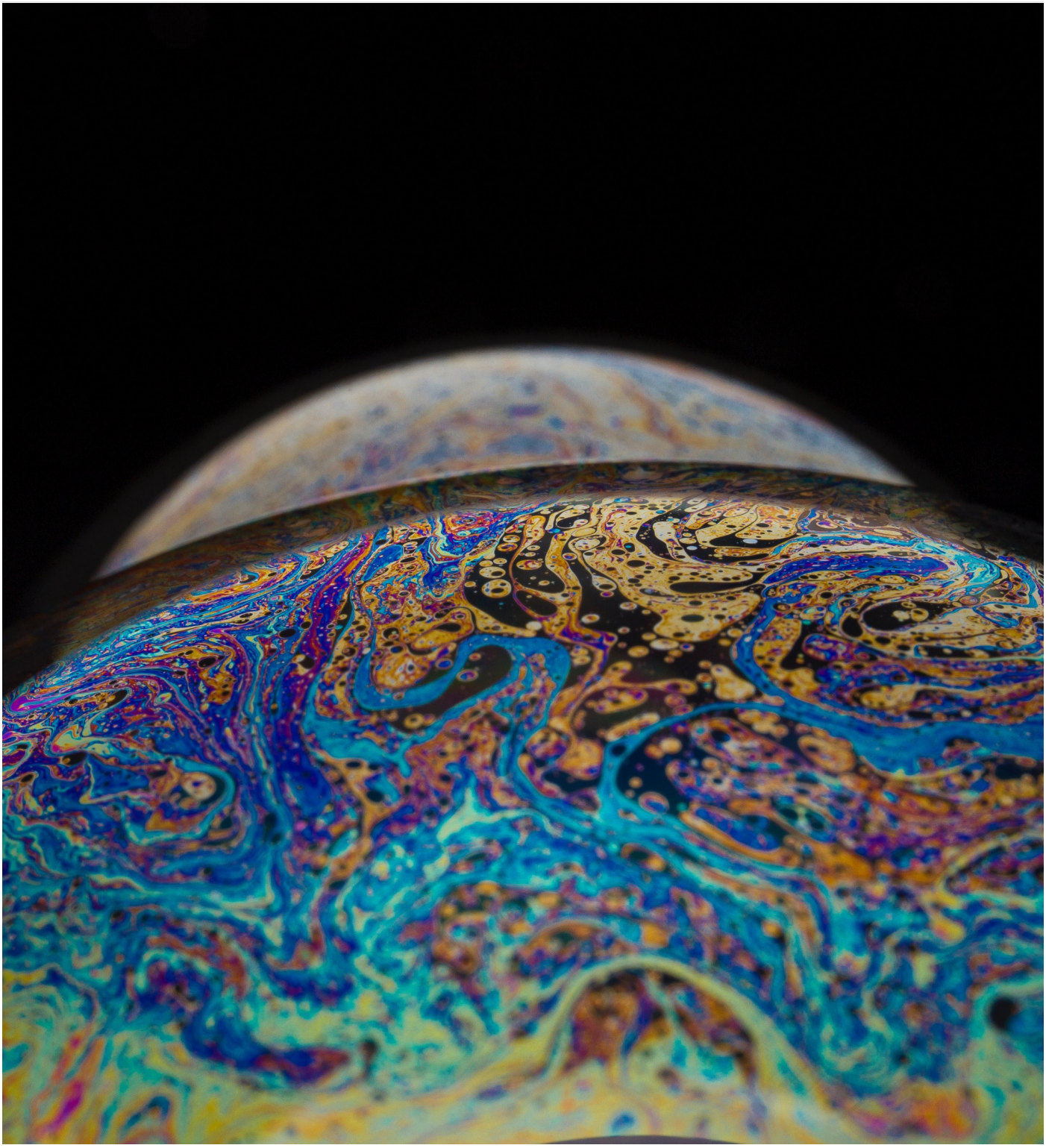

Photo by Isaac Quesada on Unsplash

The relationship between Perplexity and Entropy in NLP

Use Information Theory to understand NLP Metrics

Perplexity is a common metric to use when evaluating language models. For example, scikit-learn's implementation of Latent Dirichlet Allocation (a topic-modeling algorithm) includes perplexity as a built-in metric.

In this post, I will define perplexity and then discuss entropy, the relation between the two, and how it arises naturally in natural language processing applications.

Context

A quite general setup in many Natural Language tasks is that you have a language *L* and want to build a model *M* for the language. The "language" could be a specific genre/corpus like "English Wikipedia", "Nigerian Twitter", or "Shakespeare" or (conceptually at least) just a generic like "French."

Specifically by a language *L,* we mean a process for generating text. For clarity, we will consider the case where we are modeling sentences and the text consists of sequence words ending with an end of sentence "word." But you can replace "word" with "token" and "sentence" with "document" to generalize to any context.

What is a "process"? For our purposes, we can think of a process as a collection of probability distributions. Given a history *h* consisting of a series of previous words in a sentence, the language L is the probability that the next word is *w:*

$$h \text{ a series of words } (w_1, w_2, \ldots, w_{n-1})$$

$$L(w|h) = \text{Prob}(\text{next word is } w \mid \text{previous words are } h)$$

A language is a collection of probability distribution

For example, I am willing to wager that if L is "English":

1. L(dog | The quick brown fox jumps over the lazy brown) ≈ 1
2. L(ipsum | Lorem) ≈ 1
3. L(wings | Buffalo buffalo buffalo Buffalo buffalo) ≈ 0

Similarly, given an entire sentence *s*, we can evaluate L(*s*) the probability of the sentence occurring. If we include a special beginning of sentence "word" w_o and let

$$L(s) = L(w_1|w_0) \times L(w_2|w_0w_1) \times \cdots \times L(w_n|w_0w_1 \cdots w_{n-1})$$
$$= \prod_{k=1}^{n} L(w_k|w_0w_1 \cdots w_{k-1})$$

The Language L gives the probability of a sentence s

However it is common to leave out the first term in the product as well, or sometimes to work with an even longer starting context.

It is surprisingly easy to get a perfect replica of *L* of (say) spoken American English. Just flag down any native English speaker walking down the street. Of course, we are usually interested in teaching a computer the model (hence, Machine Learning). So we will let *M* be whatever language model we have managed to build on a computer.

This setup, with a language *L* and model *M* is quite general and plays a role in a variety of Natural Language tasks: speech-to-text, autocorrect, autocomplete, machine translation – the list goes on. Autocomplete is the most obvious example: given the words someone has typed so far, try to guess what they might type next by picking the highest-probability completion.[1]

Perplexity

Given a language model M, we can use a held-out dev (validation) set to compute the perplexity of a sentence. The perplexity on a sentence *s* is defined as:

$$\text{Perplexity}(M) = M(s)^{-1/n}$$
$$= \sqrt[n]{\prod_{k=1}^{n} \frac{1}{M(w_k|w_0w_1 \cdots w_{k-1})}}$$

Perplexity of a language model M

You will notice from the second line that this is the inverse of the *geometric mean* of

word metric. This means that, all else the same, the perplexity is not affected by sentence length.

In general, we want our probabilities to be high, which means the perplexity is low. If all the probabilities were 1, then the perplexity would be 1 and the model would perfectly predict the text. Conversely, for poorer language models, the perplexity will be higher.

It's hard to provide a benchmark for perplexity because, like most Natural Language tasks, the metric is highly dependent on the vocabulary size. Given a corpus, a smaller vocabulary means that other words will all be replaced with an <oov> (out-of-vocabulary) token, instantly increasing the apparent quality of any language model trained on it

Here are some benchmarks:

1. **State of the Art.** For WikiText-103, a selection of ~28,000 high-quality Wikipedia articles and a large (0.4% OOV rate) vocabulary, the state-of-the-art perplexity for a language model (as of this writing) is **10.8**.

2. **Worst-case-scenario.** On any dataset, the baseline model is to just guess a word in the vocabulary randomly with equal probability for each. In this case, the perplexity is just the vocabulary size: **267,735** for WikiText-103, but substantially smaller for WikiText-2 (**33,278**). 30,000, in general, is a pretty reasonable size for a language model's vocabulary.

3. **Best-case-scenario.** I said above that the "best" possible perplexity is 1. But for that to be true, there could only be one possible sentence in a language, which is quite boring.[2] A recent paper exploring text-generation uses OpenAI's GPT-2 (large version with perplexity 22.1 on WikiText-103). On their dataset of choice (WebText, which GPT-2 was trained on), they find a perplexity of **12.4**. But, crucially, they find that, while their model is capable of generating text with much lower perplexity (1.5!), the generated text is either repetitive or incoherent. Staying closer to human perplexity is better!

This last point is very important. **There is a lower bound** on perplexity fixed by the

necessarily the best predictor of the true performance of a model. Perplexity is good for development (validation) but not necessarily for evaluation. The gold standard for evaluation remains human evaluation.

Entropy

Entropy is a slippery concept in physics, but is quite straightforward in information theory. Suppose you have a process (like a language L that generates words). At each step in the process, there is some probability p that the thing that happened (the event) was going to happen. The amount of **surprisal** is $-\log(p)$ where the logarithm is taken in any base you want (equivalent to changing units). Low probability events have high surprisal. Events that were certain to happen ($p=1$) have 0 surprisals. Events that are impossible ($p=0$) have infinity surprisal.

The entropy is the expected value of the surprisal across all possible events indexed by i:

$$H(p) = -\sum_i p_i \log p_i$$

Entropy of a probability distribution p

So, the entropy is the average amount of surprise when something happens.

Entropy in base 2 is also optimal number of bits it takes to store the information about what happened, by Claude Shannon's source coding theorem. For example if I told you that a full-length tweet of 280 characters had an entropy of 1 bit per character, that means that, by the laws of mathematics, no matter what Twitter does, they will always have to have 280 bits (35 bytes) of storage for that tweet in their database. (In practice of course, they have to have quite a bit more).

In the context of our language model, we'll have to make one tweak. Given that we are interested in sentences s (sequences of events) of length n, we'll define the entropy rate per word (event) as:

$$H_n(L) = -\frac{1}{n}\sum_{s\in L} L(s)\log L(s)$$

where the sum is over all sentences of length n and L(s) is the probability of the sentence. Finally, a technical point: we want to define the entropy of the language L (or language model M) regardless of sentence length n. So finally we define

$$H(L) = \lim_{n\to\infty} -\frac{1}{n}\sum_{s\in L} L(s)\log L(s)$$

Final definition of entropy for a language (model)

The Shannon-McMillan-Breiman Theorem

Under anodyne assumptions[3] the entropy simplifies even further. The essential insight is that, if we take a long enough string of text, each sentence occurs in proportion to its probability anyways. So there is no need to sum over possible sentences. We get:

Simplification of entropy with the Shannon-McMillan-Breiman Theorem

This tells us that we can just take a large (n is big) text instead of trying to sample from diverse texts.

Cross-Entropy

Suppose we mistakenly think that our language model M is correct. Then we observe text generated by the actual language L without realizing it. The *cross-entropy* H(L,M) is what we measure the entropy to be

Cross entropy for our language model M

Where the second line again applies the Shannon-McMillan-Breiman theorem.

Crucially, this tells us we can estimate the cross-entropy H(L,M) by just measuring log M(s) for a random sample of sentences (the first line) or a sufficiently large chunk of text (the second line).

The Cross-Entropy is Bounded by the True Entropy of the Language

The cross-entropy has a nice property that $\mathbf{H(L) \leq H(L,M)}$. Omitting the limit and the normalization 1/n in the proof:

In the third line, the first term is just the cross-entropy (remember the limits and 1/n terms are implicit). The second term is the Kullback-Leibler divergence (or KL-divergence). By Gibbs' inequality the KL-divergence is non-negative and is 0 only if the models L and M are the same. The KL-divergence is sort of like a distance measure (telling you how different L and M are).[4] □

Relationship between Perplexity and Entropy

1. Evaluating the entropy of M on a sufficiently long (*n* large) set of dev/validation/test data generated by L approximates the cross-entropy H(L,M), by the Shannon-McMillan-Breiman theorem. We approximate the limit just by taking a large enough evaluation set.
2. Furthermore this cross-entropy is bounded below by the true entropy of the language generating our data

Definition of perplexity and simplification of cross-entropy for a large enough dataset

Now all that remains to do is show the relationship between the two. Assuming we took the logarithm in base e:

Relationship between perplexity and entropy

If we took the logarithm in base 2, use 2 for the base, etc.

So, to summarize:

1. We build a language model M for the true language generating the data, L.
2. We evaluate the perplexity or, equivalently, the cross-entropy of M (with respect to L).

The perplexity measures the amount of "randomness" in our model. If the perplexity is 3 (per word) then that means the model had a 1-in-3 chance of guessing (on average) the next word in the text. For this reason, it is sometimes called the *average branching factor*.

Conclusion

I want to leave you with one interesting note. It is an open question what the true entropy of written English text is (leaving aside other formats, like "Twitter" or "Spoken English" and other languages, like "Russian.")

By the inequality $H(L) \leq H(L,M)$, one way to get an upper bound on the perplexity or entropy is to create a language model, and we saw some perplexities above.

In this context, we are usually interested in the entropy *per-character* (likewise perplexity per-character). When measured using the log base 2, this becomes bits-per-character (BPC).

Claude Shannon estimated (in a time before computers) that the entropy of written English was between 0.6 and 1.3 bits per character. OpenAI's GPT-2, mentioned above, achieves about 1 bit per character on (yet another) Wikipedia dataset.

If you want to read more about information theory, see my previous article Understanding Logistic Regression Coefficients.

Keeping in mind that there are about 5 characters per word in written English, this corresponds to about 5 bits, or a perplexity of $2^5 = 32$. Note this is substantially higher than the perplexities discussed as state-of-the-art benchmarks! What gives? Remember not to compare perplexities across vocabularies or datasets: the word length may not be the same.

References

Besides those linked/mentioned above

- Speech and Language Processing (Jurafsky and Martin)
- For another take, Understanding Metrics for Language Models (Chip Huyen, The Gradient)

[1] Commonly estimated with a beam search.

for each possible starting word, or, in the context of the paper, only one possible completion of an article given the first 40 tokens. This would violate Grice's maxims, a general set of rules about language. In particular, why bother writing the rest of the article if it is determined by its beginning? Generally speaking, natural language avoids low-perplexity (entropy) utterances because they are unnecessary.

[3] The assumptions are that the process is stationary and ergodic. These assumptions do not, in fact, hold for natural language. If this bothers you, you can treat the theorem as a pretty reasonable approximation to make.

[4] It is not a distance metric because it is not symmetric D(p||q) != D(q||p). However, interpreted on a statistical manifold, its second-order Taylor expansion around D(p||p) gives the Fisher Information metric, which is the unique (up to a scalar constant, by Chentsov's Theorem) Riemannian metric suitable for statistical manifolds. See Methods of Information Geometry for further reference in the finite dimensional case.

Data Science NLP Machine Learning Artificial Intelligence