Applying NLP to Nature's Language (DNA!)

An examination of perplexity in the genetic grammar of microorganisms

Erin Wilson
NLP - CSE517 (Winter 2019)
Final Report

Abstract

Synthetic biology is a growing field that develops novel biological systems to solve pressing global challenges. One goal of synthetic biology is to enable sustainable molecule production by engineering microorganisms to produce valuable materials from renewable feedstocks, such as sugar cane, by adding foreign genes from other organisms into a host bacteria. However integrating foreign genes into a bacteria's genetic grammar can be difficult. To better understand the DNA sequence patterns involved in this genetic grammar, we pursue a language modeling approach to examine the perplexity of corpora of promoters: a key DNA signaling region. We explore a grid of k-mer n-gram model parameters as well as 4 different promoter corpora generated from either random, custom sequence extraction, or biological expert sources. We find that models of 5-mer bigrams perform the best on the most biologically precise corpus. Initial results are encouraging that language modeling is a valid way to detect patterns in genetic grammar without supervision but will require more refinement before applying this technique to organisms which lack a source of expert-curated ground truth.

Motivation

Globally, human societies are consuming finite resources at unsustainable rates. Transitioning away from our dependencies on non-renewable resources and towards a cyclical, sustainable use of natural products is critical for preserving Earth's most

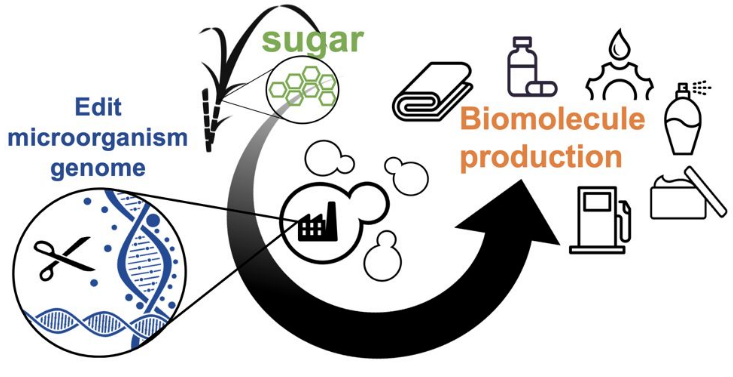


Figure 1. Microbes, such as yeast and bacteria, can be genetically engineered into factories that convert renewable resources, like sugar, into valuable biomolecules.

threatened ecosystems and securing longer term economic stability. An alternative way to source many natural products is to engineer microorganisms, such as baker's yeast or bacteria, into **biological molecule factories** [1]. After integrating foreign plant genes into the genome, the designed microorganisms can convert renewable sugar feedstocks into a wide range of valuable molecules like artemisinin (anti-malarial drug) and farnesene (jet fuel precursor) [2,3] (Figure 1). Unfortunately, the difficulty of optimizing microorganisms to efficiently use foreign genes to produce target molecules prevents many sustainably-sourced products from being cost-competitive in the market. Thus, the growth of a robust biomolecule industry has been slow.

Background

Microbe optimization remains challenging largely because it requires fine-tuning of each foreign genes' expression levels, but how does an organism know when it should turn certain genes ON and other genes OFF? Genes are defined by specific sequences of DNA letters with an alphabet of A,C,G,T. For every gene in the genome, there exist surrounding "signaling regions" that help control when a gene should be turned ON (Figure 2). These signal sequences do not actually code for the protein the gene makes, but rather they act as sorts of "flags" that sit nearby the gene. Sometimes these flags occur right before the gene, sometimes right after the gene, sometimes they even happen thousands of letters away!

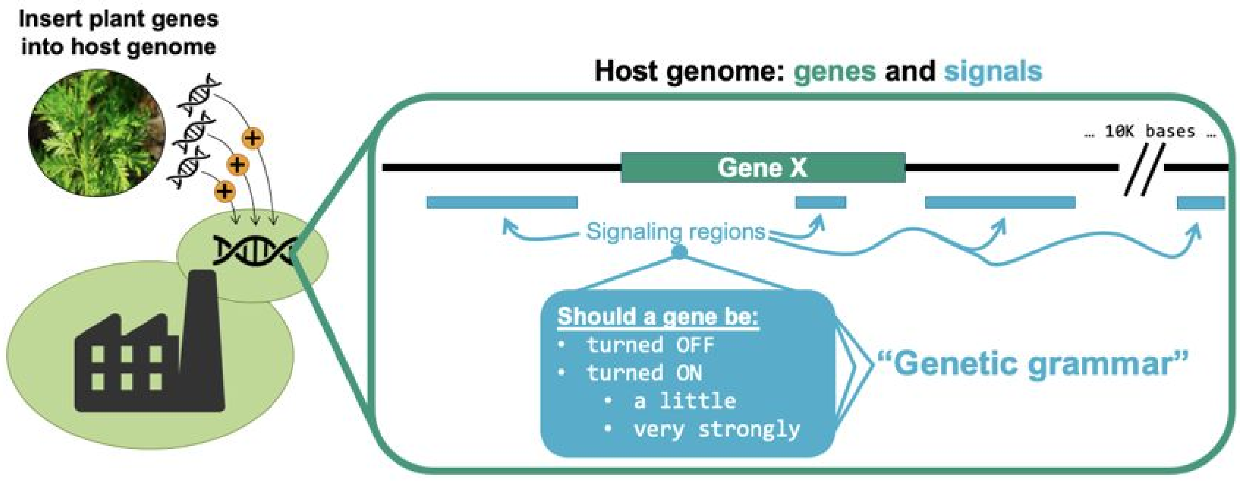


Figure 2. In addition to genes, genomes contain surrounding signaling regions with key DNA sequence patterns which make up an organism's "genetic grammar." Understanding this grammar is important when trying to express foreign genes in a bacterial system otherwise the microorganism will serve as an inefficient molecule factory.

We know generally where signaling regions are, but the exact signals and what they mean are still not well understood. For the purposes of this project, I will primarily focus on the region of DNA immediately before a gene starts, know as the **promoter**. All genes have *different* promoter sequences that control them, however related genes often share similar elements known as signalling **motifs**. Motifs are short stretches of DNA patterns, generally 4-12 bases long that can serve as the regulatory “flags.” Each promoter may have a varying combinations of these motifs that occur in different orders and in different frequencies (Figure 3). Often times these motifs can have non-linear interactions that affect gene expression.

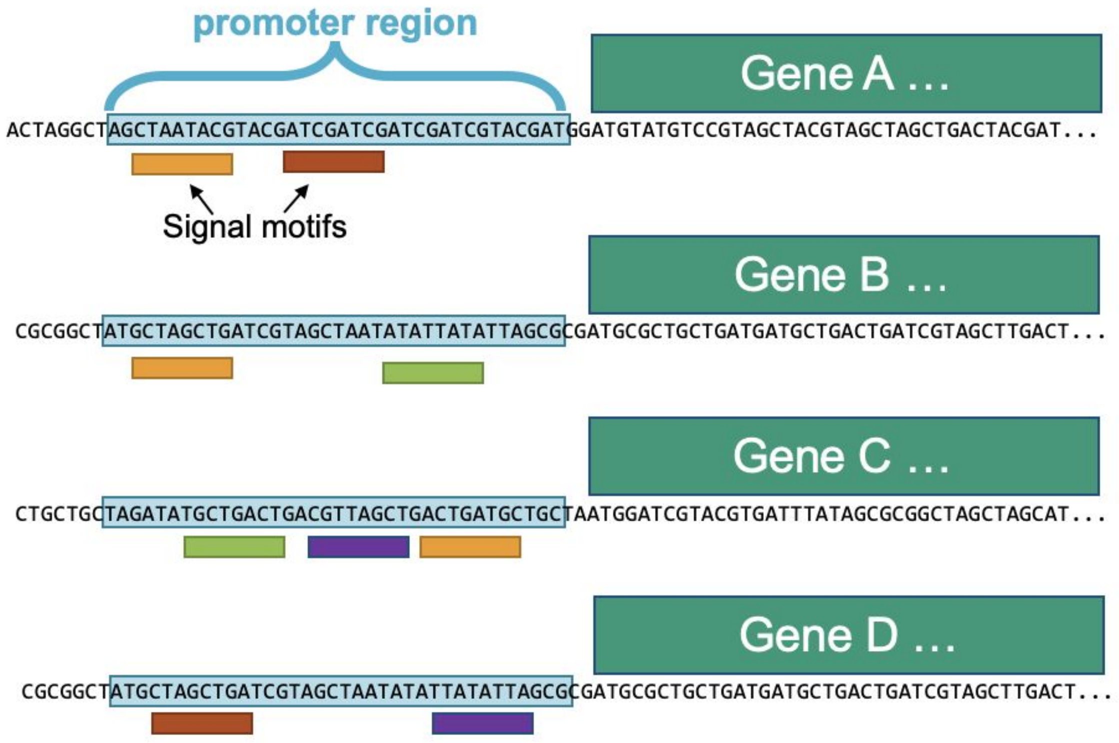

promoter region
Gene A ...
ACTAGGCTAGCTAATACGTACGATCGATCGATCGATCGTACGATGGATGTATGTCCGTAGCTACGTAGCTAGCTGACTACGAT...
Signal motifs
Gene B ...
CGCGGCTATGCTAGCTGATCGTAGCTAATATATTATATTAGCGCGATGCGCTGCTGATGATGCTGACTGATCGTAGCTTGACT...
Gene C ...
CTGCTGCTAGATATGCTGACTGACGTTAGCTGACTGATGCTGCTAATGGATCGTACGTGATTTATAGCGCGGCTAGCTAGCAT...
Gene D ...
CGCGGCTATGCTAGCTGATCGTAGCTAATATATTATATTAGCGCGATGCGCTGCTGATGATGCTGACTGATCGTAGCTTGACT...

Figure 3. Schematic of the promoter region. Every gene has a different sequence in its promoter region however these regions contain various combinations of smaller signalling elements, or motifs. (DNA sequences shown above are made up and do not correspond to real motifs.) The exact combination of motifs influences how each gene is controlled.

Related Work

Every organism has a slightly different system for regulating their genes, much like a different dialect of a language. While we understand these control systems in some organisms better than others, the exact “grammar” of DNA patterns underlying the signals is still largely unknown (eg, we cannot look at a string of 100 bases and know exactly what it will do). Years of work has been dedicated to identifying signaling motifs from biological experiments as well as computational prediction of promoter sequences from expert-curated features [4].

Given that the precise grammar behind gene regulation is still largely a mystery, I am interested in finding modeling techniques which do not use expert prior knowledge and can instead learn patterns from examples. Some work has been done to experimentally quantify 1 million randomized DNA sequences as strong or weak promoters and use convolutional neural networks to predict sequence strength [5]. But in the absence of such a large synthetic dataset for a specific organism of interest, we can also try to learn patterns directly from an organism's genome.

Approach

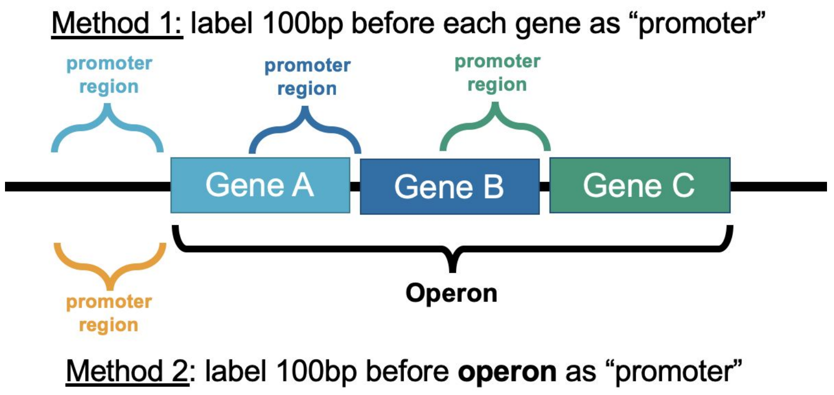

Method 1: label 100bp before each gene as "promoter"
promoter region
promoter region
promoter region
Gene A
Gene B
Gene C
Operon
promoter region
Method 2: label 100bp before operon as "promoter"

Figure 4. In many bacteria, genes are organized into operons - clusters of genes which all share one promoter. Method 2 for labeling promoter regions will more accurately characterize sequences however not all operons are explicitly labeled so it will be an approximation of likely operon clusters.

Here am I interested in 1) adapting a language modeling approach to analyze DNA sequences and 2) using perplexity to evaluate how well a language model was able to capture the genetic grammar of promoters. Specifically, if signaling regions indeed contain detectable patterns, I would expect to see lower perplexity in models trained on signalling regions versus a model trained on randomized DNA. I used 4 different corpora to train language models:

1. **Naive:** A naive approach to generating a dataset of promoters would be to pick a genome, go to each annotated gene, and slice out the sequence immediately before the gene starts and add it to a "promoter" set (Figure 4, top)
2. **Operon:** The naive dataset ignores the fact that many bacterial genes are arranged into operons, or clusters of genes which all get turned on and off together using the same promoter signal. Therefore, a better way to collect true promoter sequences might be to identify gene clusters which are likely to be in operons (eg, the genes are too close together) and only use the sequence preceding the first gene in the cluster (Figure 4, bottom). This set should be less noisy and produce a clearer language model signal.
3. **RegulonDB:** *E. Coli* is a model organism which has been extensively studied for years. RegulonDB is a database of computationally predicted promoters based on features curated by expert biologists. This promoter set is the best source of

ground truth for this organism, however most organisms to not have their promoters described in this level of detail. Therefore I'm most interested in comparing perplexities of the Operon corpus to the RegulonDB corpus. If perplexity for the Operon corpus (which I can programmatically slice from any available genome) is comparable to the perplexity of the RegulonDB set (years of work and curation), it would suggest that the language modeling approach could be robustly applied to more exotic organisms.

4. **Random:** Finally, I created a random promoter set of shuffled genomic sequences. This set would serve as a baseline since a language model should detect no patterns from these sequences.

Models & Evaluation

After collecting promoter examples as described in each of the 4 corpora listed above, next I had to decide how to break the sequences into words. DNA sequences do not have natural token boundaries like spoken languages and could be divided into arbitrary lengths of k-mer "words". Similarly, the extent of prior context used by a language model (unigram, bigram, trigram… n-gram) would differentially capture how these short sections of DNA related to each other. Therefore I created a grid of possible model parameters to try for each of the 4 corpora. In total, I trained 92 k-mer n-gram language models (Figure 5).

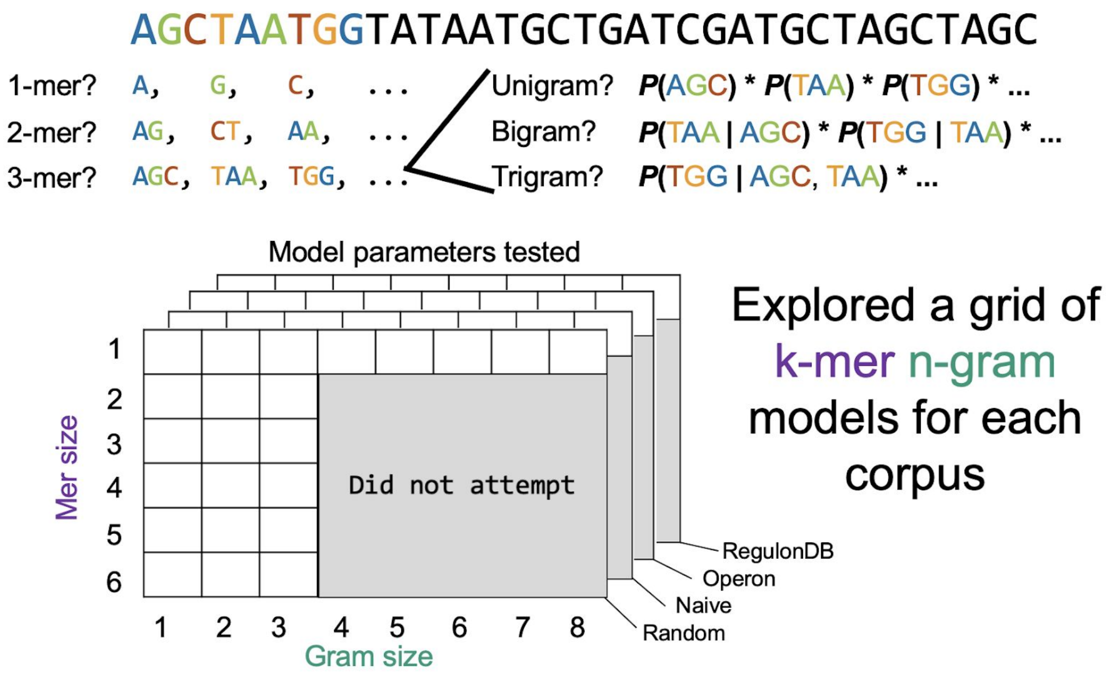

AGCTAATGGTATAATGCTGATCGATGCTAGCTAGC
1-mer? A, G, C, ...
2-mer? AG, CT, AA, ...
3-mer? AGC, TAA, TGG, ...
Unigram? P(AGC) * P(TAA) * P(TGG) * ...
Bigram? P(TAA | AGC) * P(TGG | TAA) * ...
Trigram? P(TGG | AGC, TAA) * ...
Model parameters tested
Explored a grid of k-mer n-gram models for each corpus
Did not attempt
Mer size
1
2
3
4
5
6
Gram size
1
2
3
4
5
6
7
8
RegulonDB
Operon
Naive
Random

Figure 5. Model parameter grid for testing various options of k-mer length and n-gram size. N-gram models are defined by conditional probability distributions over sequence counts.

In order to compare models to each other, I calculated the overall corpus perplexity of the test set for each k-mer n-gram setting. Briefly, perplexity is the measure of how well a model can predict a given input example given its previously calculated conditional probability distribution. It is an **inverse of the probability** of seeing an example, thus a lower perplexity indicates a better model.

Perplexity:

$$2^{-l} \text{ where } l = \frac{1}{M} \sum_{i=1}^{m} \log p(s_i)$$

Normalized Perplexity:

$$\frac{\texttt{test\_pplex}}{\texttt{worst\_possible\_pplex}}$$

- close to 1: as perplexed as can be
- 1.2: infinite perplexity

Equation 1. Calculate the perplexity of a corpus where *s* is a sequence example, *m* is the number of sequence examples, and *M* is the number of words (or k-mers) in the vocab. `Worst_possible_pplex` is 4^k.

However, because perplexity is normalized by the number of words in a vocabulary, I had to slightly adjust my perplexity evaluation of DNA corpora. Typically when comparing the perplexity between two models, it is common for both models to have used the same corpus (eg, the Brown corpus) and therefore the same number of words in the vocabulary. For this project, depending on my choice of k-mer size, the size of my "possible" vocab also changed (4^k). To make perplexity comparisons between models easier, I normalized all raw perplexity score to 4^k (the worst possible perplexity besides infinity). Consequently, normalized perplexity scores around 1.0 can be interpreted as "as perplexed as can be" while lower than 1 indicates better than random perplexity.

In cases where an infinite perplexity occured due to model sparsity (e.g. seeing a probability of 0 for a given k-mer in the test set), I automatically set the normalized perplexity to 1.2 so that all the models could be plotted in a heatmap on the same Viridis color scale. This ensured that 1.2 would appear at a significantly darker purple than the medium blue hue at 1.0 and that differences in lower model perplexities were discernible. Without setting infinite scores to 1.2, the rest of the models would be pushed to the extreme yellow end of the Viridis scale and be indistinguishable (Supplementary Figure 1).

However infinite perplexities are undesirable because it means your model cannot generalize well to new contexts. To combat this, smoothing techniques should be applied. For this project I attempted implementations at Laplace smoothing as well as the more general add-k smoothing. Unfortunately these techniques require further debugging and thus the results presented below are un-smoothed. (See discussion in the *Supplementary Information - Smoothing* for more details.)

Results

After training 92 language models across 4 different corpora and calculating the normalized perplexities, a few encouraging patterns emerged. First, I considered how different choices of k and n influence perplexity and visualized the results as a heatmap. Across all models in the Random Corpus, perplexity scores were near 1.0 until the selected values of k and n resulted in too sparse of a corpus, causing infinite perplexities (e.g. 6-mer bigram, 4-6-mer trigram models). This suggested that the language model could detect no patterns in random DNA, as expected (Fig 6A).

For the more biologically relevant corpora, we see a significant deviation from random. As k and n increase, we see a drop in perplexity, suggesting that k-mer sizes of about 3-5 are better than 1-2 and that bigram and trigram models perform better than unigram models. Similarly, 6-mer bigram and 4-6-mer trigram models were too sparse. The best perplexities occured in the RegulonDB corpus for the 5-mer bigram model (Figure 6C).

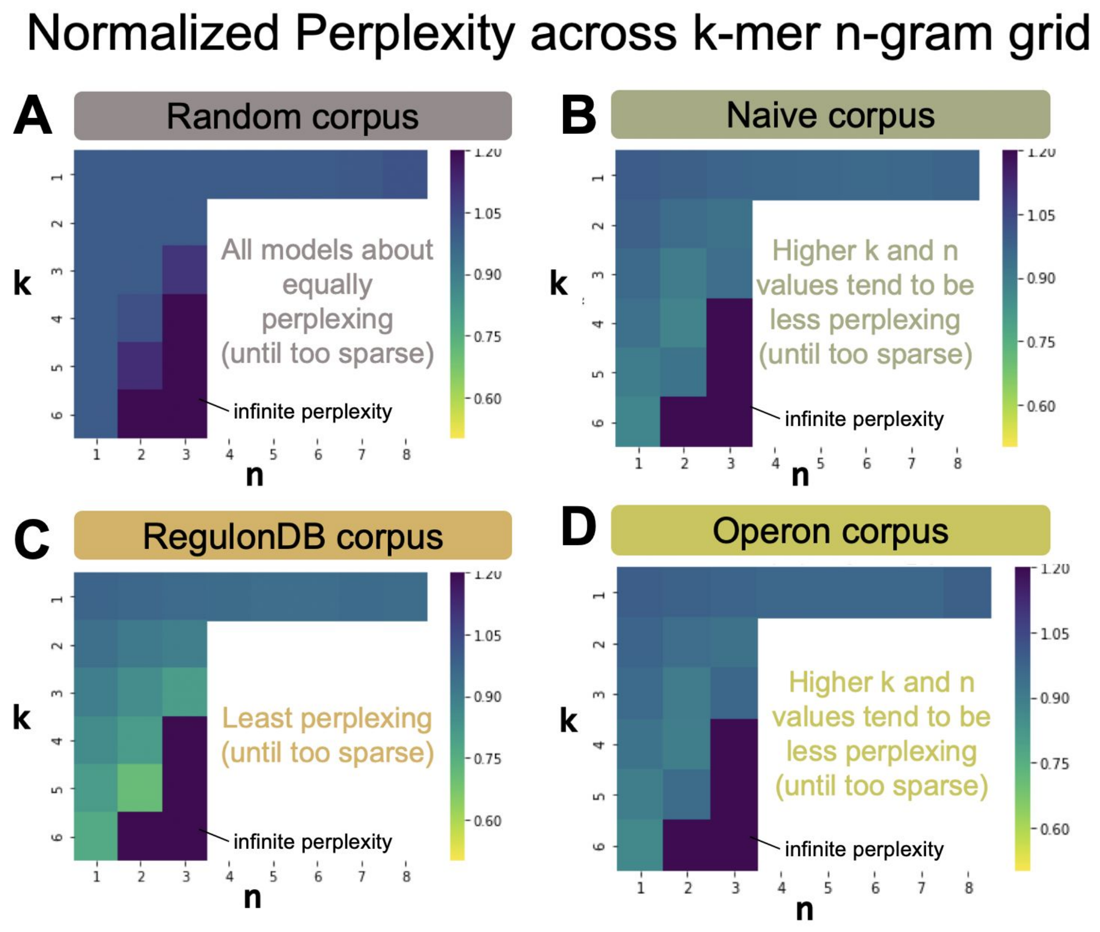


Figure 6. Normalized perplexity for various combinations of k and n model parameters. Values of 1.2 (dark purple) represent infinite perplexity, 1.0 (blue) represents worst perplexity, and light colors represent lower perplexity.

Looking at the same data from a different angle, we can better compare corpora to each other by visualizing a series of bar charts (Figure 7). The key trend from these visuals is that across most model choices of k and n, we see a decreasing trend in perplexity as the “biological precision” of the promoter set increases (Random >> Naive >> Operon

>> RegulonDB). Surprisingly, there are a few cases where the Operon perplexity is *higher* than Naive (e.g. Figure 7, n=2, k=5). In all cases, RegulonDB performs the best.

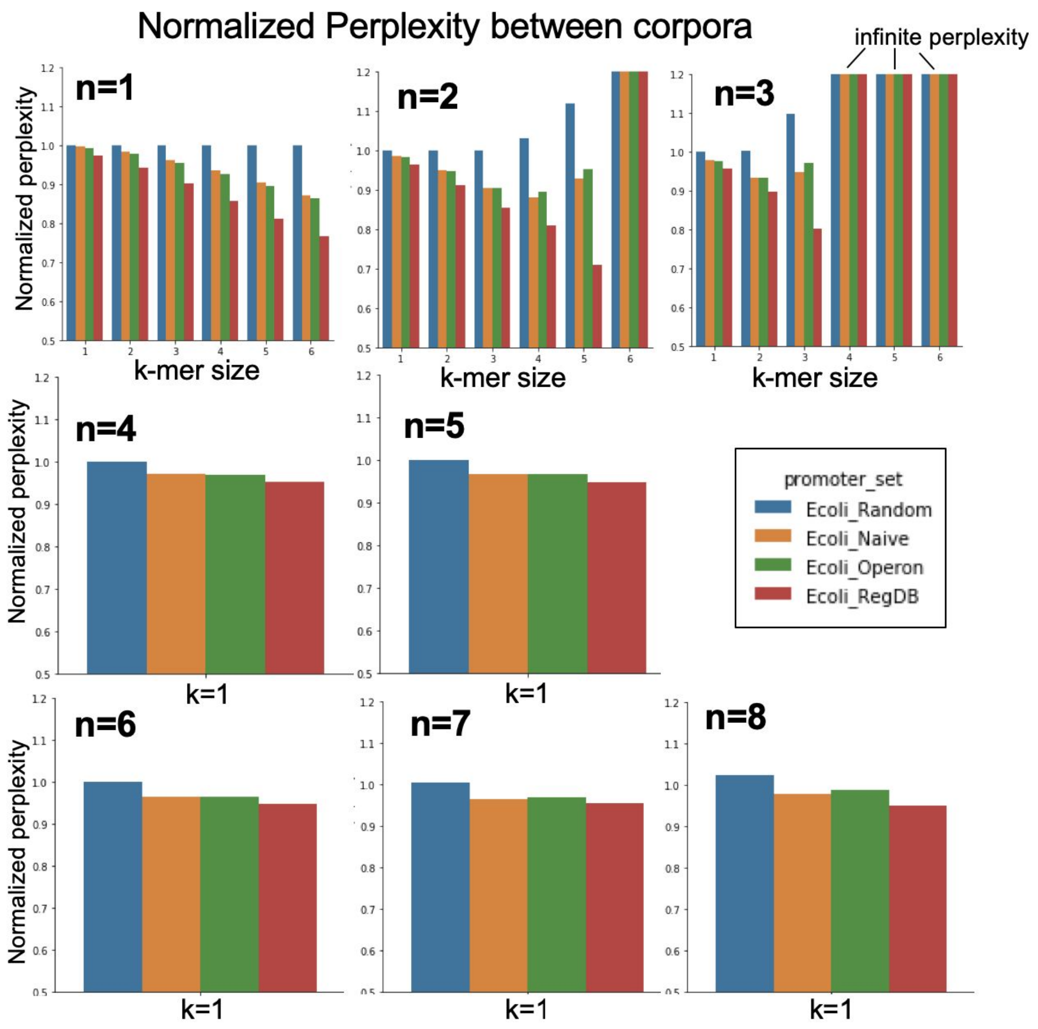


Figure 7. Normalized perplexity across each promoter corpus. Corpora are ordered by "biological relevance" (e.g. Random has no promoter signal, Naive and Operon should have basic promoter signal, and RegulonDB is the current best source of ground truth for what real *E. Coli* promoters look like.)

Discussion

The results from this project suggest that language modeling can indeed capture patterns that exist in promoter signaling regions that do not exist in randomized sequences. Especially encouraging is that in most cases, the Operon corpus yields a lower perplexity than the Naive corpus, which likely contains erroneous example "promoter" sequences which are actually part of other genes and not promoters (Figure 4).

The best choice of k and n for the k-mer n-gram modeling appear to be the 5-mer bigram model or the 3-mer trigram model. In biology, motif signals tend to range from 4-12 letters and both of these models offer contextual information of about 10 and 9 consecutive letters, respectively. Additionally, since signal motifs can be fuzzy (e.g. TATAAA vs TATGAA may be similar signals but with slightly different strengths), breaking up the promoter examples into 3-5-mers may offer the flexibility to be able to probabilistically capture slight differences between similar signals.

From the plots in Figure 7, it is clear that the model trained on the Operon corpus is still systematically worse than the same model parameters trained on RegulonDB. Ideally, if these perplexity scores were to be more similar, it would suggest that the language model could as accurately capture the patterns in regulatory signals without the reliance on expert-curated features. In that case, I would be more confident applying this modeling technique to less well-studied organisms, for example the methane-eating bacteria my advisor works with, *Methylomicrobium buryatense* (a methanotroph, Figure 8). At this point however, the RegulonDB promoter set seems to have a clearer signal that the language models are able to pick up. Therefore the current method for obtaining an Operon corpus would need more refinement so that it can match the perplexity performance of the higher quality promoter set before extending to other organisms.

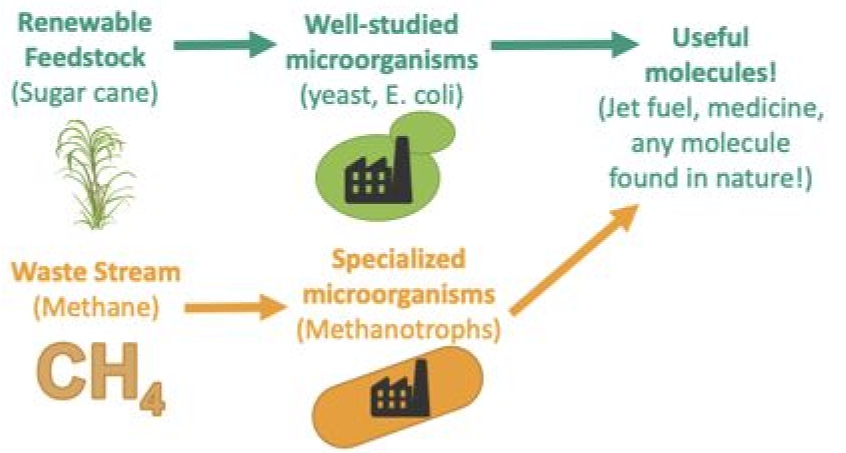

Renewable Feedstock (Sugar cane)
Well-studied microorganisms (yeast, E. coli)
Useful molecules! (Jet fuel, medicine, any molecule found in nature!)
Waste Stream (Methane)
CH_4
Specialized microorganisms (Methanotrophs)

Figure 8. An alternative paradigm for sustainable molecule production. By learning the genetic grammar of bacteria like methanotrophs, instead of using renewable sugarcane as a feedstock for an engineered microorganism factory, we can instead use methane as a feedstock. Methane is an abundant human waste stream and a greenhouse gas ~20x more potent than CO2. Diverting methane into a process which converts waste into a product of value is a promising way to source natural molecules even *more* sustainably than from sugar cane!

Future work

In the future, I would like to continue investigating DNA perplexity, but rather than summarizing perplexity for an entire corpus, I'd likely to calculate perplexity across each sequence. Ideally, this type of analysis should be able to indicate sequence regions of low perplexity which should contain the “truest promoter signal.” Some preliminary results suggest that indeed there can be regions within a sequence with differential perplexity.

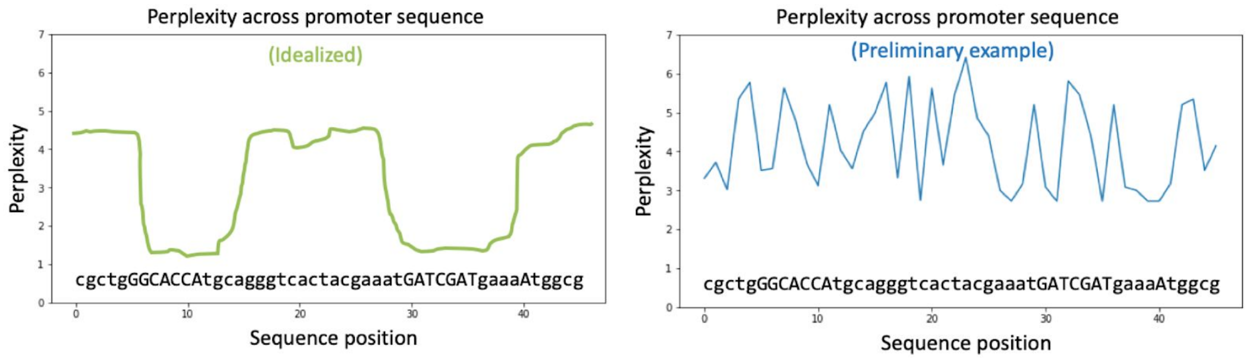


Figure 9. Preliminary calculations of perplexity across a DNA sequence. Instead of averaging the perplexity score for each sequence, I plot the individual perplexity scores at each position. Regions which the model is well trained to predict (and therefore are likely to contain key elements of the genetic grammar) should appear to have lower perplexity. *Left:* idealized version with two clear dips in perplexity at key signal regions. *Right:* an example perplexity vector from a k=1, n=1 model for the *apaH* promoter.

Furthermore, one could imagine smoothing the perplexity vector itself by taking chunks of perplexity averages across the sequences (a coarser summary than every single position) [idea credit: Max!]. Efforts to determine how best to summarize low perplexity regions across models and ultimately extract the “true signal” of a sequence is on-going work.

Conclusion

This project applied language modeling techniques from Natural Language Processing to DNA sequences in order to better understand the “genetic grammar” underlying signalling regions in microorganisms. DNA “word” sizes of 3-5 and bi/trigram models appeared to be the best model parameters across the 3 biologically relevant promoter corpora and yield the best perplexity scores for the expert-curated RegulonDB promoter set. More refinement is needed in order to apply these techniques to other organisms, such as the methane-eating bacteria *Methylomicrobium buryatense.*

References

[1] A. L. Meadows et al., “Rewriting yeast central carbon metabolism for industrial isoprenoid production,” *Nature*, vol. 537, no. 7622, pp. 694–697, Sep. 2016.

[2] D.-K. Ro et al., “Production of the antimalarial drug precursor artemisinic acid in engineered

yeast," *Nature*, vol. 440, no. 7086, pp. 940–943, Apr. 2006.

[3] K. R. Benjamin et al., "Developing Commercial Production of Semi-Synthetic Artemisinin, and of β-Farnesene, an Isoprenoid Produced by Fermentation of Brazilian Sugar," *J. Braz. Chem. Soc.*, vol. 27, no. 8, pp. 1339–1345, Aug. 2016.

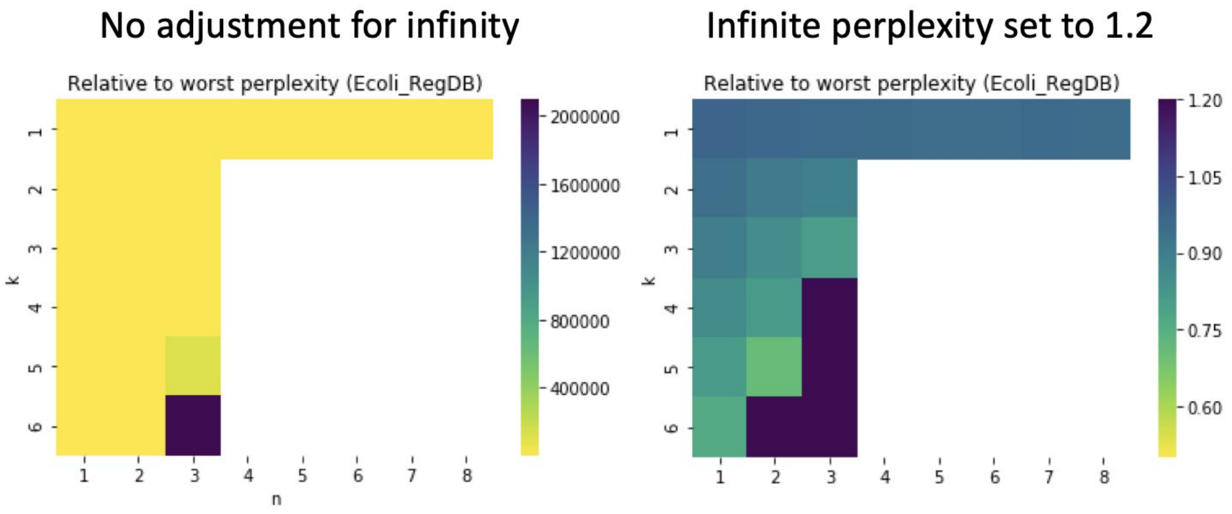
[4] A.M. Huerta, J. Collado-Vides. "Computational Prediction of Promoters in the Escherichia coli genome." *J Mol Biol.* 2003.

[5] J. T. Cuperus et al. "Deep learning of the regulatory grammar of yeast 5′ untranslated regions from 500,000 random sequences." *Genome Research.* 2017.

Supplementary Material

Normalized Perplexity

When normalizing perplexity scores to the worst possible perplexity (4^k), it was visually helpful to set an automatic catch to convert infinity to 1.2.

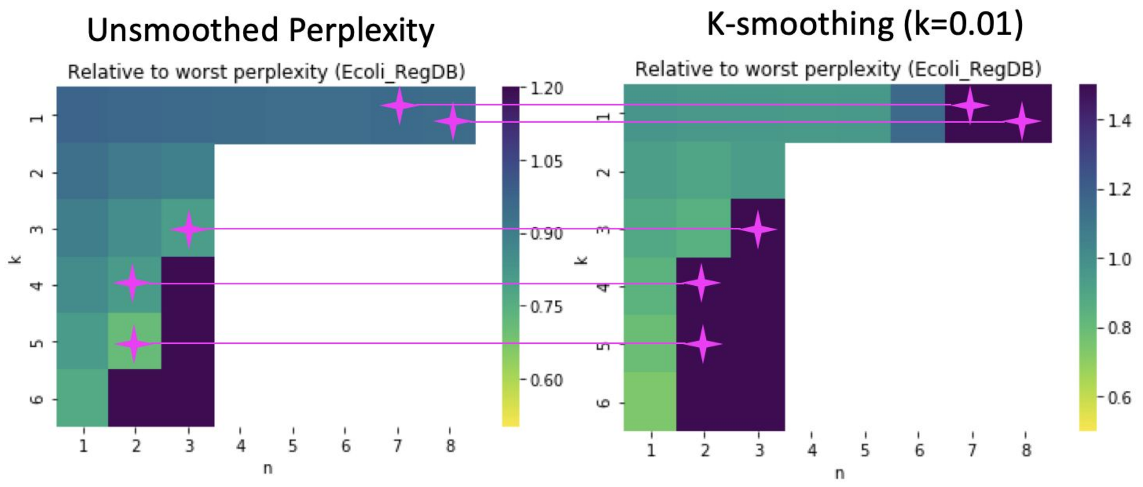


Supplementary Figure 1. Without automatically catching infinite perplexity and setting it to 1.2, the normalized perplexity washes out the color Viridis scale. The catch, though somewhat arbitrary, allows all the values to be discriminable at lower perplexity while the infinite values remain obvious.

Smoothing

For greater values of k and n, the number of example n-grams got increasingly sparse. Whenever a k-mer appeared in the test set that was not in the training set, it would result in a perplexity of infinity (probability of 0). In class we discussed several methods for smoothing, including Laplace (add 1) smoothing and the more general add-k smoothing. My initial implementation of Laplace smoothing attempted to pretend that we saw every combination of k-mer n-grams 1 more time than we actually did. For values of k=6 and n=3, this would add $4^{(k*n)}$ (~70 billion) entries to the table, which was computationally too expensive.

After helpful discussions with a great TA ;) I tried an alternative smoothing implementation which instead of adding 1 to every combination and storing in a dictionary, instead tried to handle the smoothing by adding k at compute time (when processing the test set). While this method was much more computationally tractable, there still seemed to be a problem with the implementation. Specifically, while perplexity scores were no longer infinity, they were much worse than the theoretical worst perplexity (normalized values much > 1). Furthermore, on models where the perplexity was fine without smoothing, the addition of smoothing seemed to make it worse (Supplementary Figure 2, pink stars).



Supplementary Figure 2. RegulonDB perplexity grid with and without k smoothing (k=0.01). Several models which had reasonably low normalized perplexities without smoothing suddenly became “worse than the worst possible” perplexity after smoothing. Something is clearly in need of debugging!

After further debugging, hopefully add-k smoothing should alleviate the infinite perplexity problem. Practically speaking, the 6-mer trigram models are likely towards the extreme end of how long a signal is likely to be (18 bases) and so perhaps these models are not as critical for interpreting true biological relevance.

Final Entertaining Image for the TA!