Research Blog (/blog/)

Maximum Likelihood Decoding with RNNs - the good, the bad, and the ugly

Russell Stewart (http://russellsstewart.com) 04/26/2016

Training Tensorflow's large language model on the Penn Tree Bank yields a test perplexity of 82. With the code provided here (https://www.tensorflow.org/versions/r0.8/tutorials/recurrent/index.html), we used the large model for text generation, and got the following results depending on the temperature parameter used for sampling:

$\tau = 1.0$

> The big three auto makers posted a N N drop in early fiscal first-half profit. The same question is how many increasing cash administrative and financial institutions might disappear in choosing. The man in the compelling future was considered the city Edward H. Werner Noriega's chief financial officer were unavailable for comment.

$\tau = 0.5$

> The proposed guidelines are expected to be approved by the end of the year. The company said it will sell N N of its common shares to the New York Stock Exchange. The New York Stock Exchange's board approved the trading on the big board to sell a N N stake in the company.

Which sample is better? It depends on your personal taste. The high temperature sample displays greater linguistic variety, but the low temperature sample is more grammatically correct. Such is the world of temperature sampling - lowering the temperature allows you to focus on higher probability output sequences and smooth over deficiencies of the model. But if you dig into the math, there's actually a lot more going on.

The freezing function

Temperature sampling works by increasing the probability of the most likely words before sampling. The output probability p_i of each word is transformed by the freezing function f to:

$$\tilde{p}_i = f_\tau(p)_i = \frac{p_i^{\frac{1}{\tau}}}{\sum_j p_j^{\frac{1}{\tau}}}$$

For $\tau = 1$, the freezing function is just the identity function. For $\tau \rightarrow 0$, the freezing function turns sampling into the argmax function, returning the most likely output word. For $\tau = 0.5$, the freezing function is equivalent to squaring the probability of each output word, and then renormalizing the sum of probabilities to 1. The typical perspective I hear is that a temperature like 0.5 is supposed to make the model more robust to errors while maintaining some diversity that you'd miss out on with a greedy argmax sampler.

But what if our model was fantastic and didn't make any errors? What would the effect of temperature sampling be in that case? If we look at a simple grammar where an LSTM won't make any mistakes, we can start to answer this question.

What day of the week is it?

Suppose your are asked what day of the week it is, and you have a 70% chance of knowing the answer. 30% of the time you respond "I don't know". The remaining answers of "Monday", "Tuesday", etc. each occur with probability 10%. Your responses are over a few months and you want to train a Recurrent Neural Network to generate your responses. Given the simplicity of the task, the neural network will learn the probability of each answer with high precision, and won't be expected to make any errors. If you use $\tau = 1.0$, you'll get representative samples from the same 70/30 distribution in which you uttered them.

But if you use $\tau = 0.5$, will the network be more or less likely to know what day of the week it is? Temperature sampling biases samples towards more likely responses, but in this case, lowering the temperature will actually cause the chance that you know the answer to go down! Squaring the probability for each specific answer and renormalizing yields $\tilde{p}$ with a 6.25% chance of answering "Monday", "Tuesday", etc., and a 56.25% chance of responding "I don't know". Maybe you think that this is an okay result. After all, "I don't know" was the single most likely response. But there is a different perspective under which we should expect the probability of the network knowing the day of the week to have gone up.

What if instead of recording your answers verbatim, you had recorded your responses as simply knowing or not knowing what day of the week it was? We could go back and replace each instance of "Monday" or "Tuesday" etc. in the training set with "I do know". After training that model, temperature sampling with $\tau = 0.5$ would causes the probability that the network knows the day of the week to go *up* to 84.5%. To remove any vocabulary changes, we could further go back and sample the day of the week you answered whenever you responded "I do know", and produce each answer of "Monday", "Tuesday", etc. with probability 12.1%. "I don't know" would be produced 14.5% of the time.

Sampling for the semantic category before sampling at the word level decreases the probability of the "I don't know" response

Semantic temperature sampling

Which of these two sampling methods is correct? Both have natural interpretations, but they give completely different results. In some cases, the latter two-stage sampling method may be more appropriate, and we define it formally here. Given two temperatures τ_1 and τ_2, and a semantic partition $\phi : \text{words}\epsilon[1..N] \rightarrow \text{categories}\epsilon[1..k]$, we define the semantic freezing function $h_{\tau_1,\tau_2,\phi}$ as follows:

$$
\begin{aligned}
q_j &= \textstyle\sum_i p_i * 1\{\phi(i) == j\} \\
\tilde{q_j} &= f_{\tau_1}(q)_j \\
r_i^j &= Pr[i|\text{category} = j] = \tfrac{p_i * 1\{\phi(i)==j\}}{q_j} \\
\tilde{r}^j &= f_{\tau_2}(r_i^j) \\
\tilde{p_i} &= h(p)_i = \tilde{q}_{\phi(i)} * \tilde{r}_i^{\phi(i)}
\end{aligned}
$$

That is, we partition our vocabulary into k semantic categories. At each output step, we compute the probability that the output word is in each category, and sample from this distribution with temperature τ_1. Once a category is decided, we sample among words within that category with temperature τ_2. Note that semantic temperature sampling generalizes classical temperature sampling as we may choose to use only a single semantic category and let $\tau_2 = \tau$ (i.e. $f_\tau(p) = h_{1,\tau,\text{lambda } i:1}(p)$). Alternatively, we may also chose N discrete categories, one for each word, and let $\tau_1 = \tau$ (i.e. $f_\tau(p) = h_{\tau,1,\text{lambda } i:i}(p)$).

Returning to our original example, what kind of output do we get from semantic temperature sampling? We define ϕ by running k-means with 100 categories on the word vector projection matrix, and then sample as:

$\tau_1 = 0.5, \tau_2 = 1.0$

> The vague tone of the Seattle business has been first to be offset by the Oct. N massacre in the state. The president said that when the bank leaves economic development of a foreign contractor, it is not offered to find a major degree for the market. The Miller Metal Co. unit of the national airline, which publishes the caribbean and its latest trading network, will be the first time since the new company has completed the sale of New York City Bank in Pittsburgh.

In the above, we're heavily weighting the most likely categories, but then backing off and sampling less aggressively with $\tau_2 = 1.0$ within a category. It's not clear that this output is *better* than any that could be achieved with traditional temperature sampling. Achieving lower perplexity on the Penn Tree Bank would be more impactful to that end. But we do see qualitative changes in the output when turning the new τ_1 knob. The sampling regime above focuses on the stock market semantics so frequently found in the Wall Street Journal without overusing individual terms like "company" and "New York Stock Exchange" as in the original example.

Maximum likelihood decoding

Armed with the tool of semantic temperature sampling, we can make a few more interesting connections within the realm of RNN decoding. Consider the case where both $\tau_1 \rightarrow 0$ and $\tau_2 \rightarrow 0$. This decoding scheme corresponds to first picking the most likely semantic category, and then picking the best way of expressing those semantics. If the scheme fully achieved this claim, it would be quite satisfying. But the semantic categories we're thinking of here are constrained to be at the word level, not the sentence level. In our day of the week example above, these happen to be identical, but that need not be the case in general.

If the semantics do need to take place at the sentence level, there is no clear path forward in the general case. While we may use k-means as an attempt at word level semantics, it's unclear what kind of systematic strategies could be used for sentence level clustering. One could try sentence vectors, but those are not directly available from the task at hand. The idea of a sample that first 1) figures out what semantics to respond with and then 2) figures out how to express those semantics is a nice abstraction. But word-level semantic temperature sampling as defined above only gives us an approximation.

What then are we to do? LSTM language models trained end-to-end give us a beautiful abstraction; minimizing perplexity on the training set produces an optimal word sampler for free. But if we want a maximum likelihood decoder, we have to define semantics and we're in trouble. If we don't define semantics, we'll just implicitly be assuming that all words have their own independent semantic category [1]. In the simplest case where word level semantics suffice, we can provide a ϕ function to the semantic temperature sampler. Sadly though, the choice of ϕ is not quantitatively justified, as it is not encoded in the training loss. As a result, the very idea of maximum likelihood sampling from a perplexity-trained language model is still somewhat dubious.

Conclusion

Temperature sampling is a standard technique for improving the quality of samples from language models. But temperature sampling also introduces semantic distortions in the process. We explored these distortions in the context of a simple grammar, and introduced semantic temperature sampling as a method to control them through the semantic function ϕ. We fall short of defining a meaningful objective function over which to compare different sampling regimes, and punt on a metric for comparing choices of ϕ.

Humans can disambiguate the advantages of varied sampling schemes because their conversational responses are ultimately derived from the evolutionary advantages of strong communication. Such an evolutionary pressure would likewise provide a principled objective function for machine conversational semantics in the general case.

Acknowledgments

Thanks to Chris Manning (http://nlp.stanford.edu/manning/) for advising this research. Thanks to Jiwei Li (http://web.stanford.edu/~jiweil/), Thang Luong (http://www.cs.stanford.edu/~lmthang/), Andrej Karpathy (http://cs.stanford.edu/people/karpathy/), Tudor Achim (http://www.cs.stanford.edu/~tachim/), and Ankit Kumar (http://stanford.edu/~ankitk/) for providing insightful feedback.

Notes

[1] Imagine inserting the alias "zn" for the word "an" throughout the corpus in 50% of "an" instances. How would this impact a maximum likelihood decoder trained on that corpus? Hint: how would this affect the ability of the "an" token to compete with the "a" token in the maximum likelihood sense? This one simple change could significantly decrease the presence of all words beginning with vowels in our samples.

Stanford NLP Group

Gates Computer Science Building
353 Serra Mall
Stanford, CA 94305-9020
Directions and Parking (http://forum.stanford.edu/visitors/directions/gates.php)

Affiliated Groups

- **Stanford AI Lab (http://ai.stanford.edu/)**
- **Stanford InfoLab (http://infolab.stanford.edu/)**
- **CSLI (https://www-csli.stanford.edu/)**

Connect

- **Stack Overflow (http://stackoverflow.com/tags/stanford-nlp)**
- **Github (https://github.com/stanfordnlp/CoreNLP)**
- **Twitter (https://twitter.com/stanfordnlp)**

Local links

NLP lunch (/local/nlp_lunch.shtml) · NLP Reading Group (http://nlp.stanford.edu/read/)
NLP Seminar (http://nlp.stanford.edu/seminar/) · Calendar (/local/calendar.shtml)
JavaNLP (/javanlp/) (javadocs (/nlp/javadoc/javanlp/)) · machines (/local/machines.shtml)
AI Speakers (http://ai.stanford.edu/portfolio-view/distinguished-speaker-series) · Q&A (/local/qa/)