Self-Supervised Language Modeling

CSCI 601 471/671
NLP: Self-Supervised Models

https://self-supervised.cs.jhu.edu/sp2023/

[Slide credit: Mohit Iyyer, Chris Manning, and many others]
Motivation

- Earlier we define **Self-Supervised** models as predictive models of the world!

- **Word2Vec**: Predictive models given word representations.

- **Now with neural networks**: predictive models of word compositions
Chapter Plan

1. Language modeling: definitions and history
2. Language modeling with counting
3. Measuring language modeling quality
4. Language Modeling with feed-forward networks
The cat
The cat sat
The cat sat on
The cat sat on ___?___
The cat sat on the mat.
The cat sat on the mat.
$P(\text{mat} \mid \text{The cat sat on the})$
Probability of Upcoming Word

\[ P(X_t \mid X_1, \ldots, X_{t-1}) \]

- next word
- context or prefix
LMs as a Marginal Distribution

- Directly we train models on “marginals”:

\[ P(X_t | X_1, \ldots, X_{t-1}) \]

“The cat sat on the [MASK]”

Some model
LMs as Implicit Joint Distribution of Language

- Though implicitly we are learning the full distribution over the language:
  - Remember the chain rule: $P(X_1, ..., X_t) = P(X_1) \prod_{i=1}^{t} P(X_i | X_1, X_2, ..., X_i)$

- **Language Modeling** $\triangleq$ learning prob distribution over language sequence.
Doing Things with Language Model

- What is the probability of ....
  “I like Johns Hopkins University”
  “like Hopkins I University Johns”

- LMs assign a probability to every sentence (or any string of words).

\[
P(“I \text{ like Johns Hopkins University EOS”}) = 10^{-5}
\]

\[
P(“\text{like Hopkins I University Johns EOS”}) = 10^{-15}
\]
Doing Things with Language Model (2)

- We can rank sentences.

- While LMs show “typicality”, this may be a proxy indicator to other properties:
  - Grammaticality, fluency, factuality, etc.

\[
P(X_t | X_1, \ldots, X_{t-1})
\]

\[
P(“I like Johns Hopkins University. EOS”) > P(“I like John Hopkins University EOS”)\]
\[
P(“I like Johns Hopkins University. EOS”) > P(“University. I Johns EOS Hopkins like”)\]
\[
P(“JHU is located in Baltimore. EOS”) > P(“JHU is located in Virginia. EOS”)\]
Doing Things with Language Model (3)

- Can also generate strings
- Let’s say we start “Johns Hopkins is ”
- Using this prompt as initial condition, recursively sample from an LM:

1. Sample from $P(X_t | X_1, \ldots, X_{t-1})$ → “located”
2. Sample from $P(X_t | X_1, \ldots, X_{t-1})$ → “at”
3. Sample from $P(X_t | X_1, \ldots, X_{t-1})$ → “the”
4. Sample from $P(X_t | X_1, \ldots, X_{t-1})$ → “state”
5. Sample from $P(X_t | X_1, \ldots, X_{t-1})$ → “of”
6. Sample from $P(X_t | X_1, \ldots, X_{t-1})$ → “Maryland”
7. Sample from $P(X_t | X_1, \ldots, X_{t-1})$ → “EOS”
Why Should We Care About Language Modeling?

- Language Modeling is an effective proxy for language understanding.
  - Effective ability to predict forthcoming words rely on understanding of context/prefix

- Language Modeling is a subcomponent superset of many NLP tasks, especially those involving text generation:
  - Summarization
  - Machine translation
  - Spelling correction
  - Dialogue etc.
You use Language Models every day!
You use Language Models every day!
You use Language Models every day!

Welcome to ChatGPT

Hi

Hello! How can I help you today? Is there something you need help with or would like to learn more about? I'm here to assist you with any questions you may have.
It Can be Misused Too ...

- A lot more about harms later in the class.

Is this a real science article?

Rooter: A Methodology for the Typical Unification of Access Points and Redundancy

Jeremy Stribling, Daniel Aguayo and Maxwell Krohn

ABSTRACT

Many physicists would agree that, had it not been for congestion control, the evaluation of web browsers might never have occurred. In fact, few hackers worldwide would disagree with the essential unification of voice-over-IP and public-private key pair. In order to solve this riddle, we confirm that SMPs can be made stochastic, cacheable, and interposable.

I. INTRODUCTION

Many scholars would agree that, had it not been for active networks, the simulation of Lamport clocks might never have occurred. The notion that end-users synchronize with the investigation of Markov models is rarely outdated. A theoretical grand challenge in theory is the important unification of virtual machines and real-time theory. To what extent can web browsers be constructed to achieve this purpose?

Certainly, the usual methods for the emulation of Smalltalk that paved the way for the investigation of rasterization do not apply in this area. In the opinions of many, despite the fact that conventional wisdom states that this grand challenge is continuously answered by the study of access points, we believe that a different solution is necessary. It should be noted that Rooter runs in $\Omega(\log n \log \log n)$ time. Certainly, the shortcoming of this type of solution, however, is that compilers and saperpages are mostly incompatible. Despite the fact that similar methodologies visualize XML, we surmount this issue without synthesizing distributed archetypes.

The rest of this paper is organized as follows. For starters, we motivate the need for fiber-optic cables. We place our work in context with the prior work in this area. To address this obstacle, we disprove that even though the much-touted autonomous algorithm for the construction of digital-to-analog converters by Jones [10] is NP-complete, object-oriented languages can be made signed, decentralized, and signed. Along these same lines, to accomplish this mission, we concentrate our efforts on showing that the famous ubiquitous algorithm for the exploration of robots by Sato et al. runs in $\Omega((n + \log n))$ time [22]. In the end, we conclude.

II. ARCHITECTURE

Our research is principled. Consider the early methodology by Martin and Smith; our model is similar, but will actually overcome this grand challenge. Despite the fact that such a claim at first glance seems unexpected, it is buffeted by previous work in the field. Any significant development of secure theory will clearly require that the acclaimed real-time algorithm for the refinement of write-ahead logging by Edward Feigenbaum et al. [15] is impossible; our application is no different. This may or may not actually hold in reality.

We consider an application consisting of $n$ access points. Next, the model for our heuristic consists of four independent components: simulated annealing, active networks, flexible modalities, and the study of reinforcement learning.

We consider an algorithm consisting of $n$ semaphores. Any unproven synthesis of introspective methodologies will

https://pdos.csail.mit.edu/archive/scigen/
How do we estimate these probabilities?
Let’s just count!

\[ P(X_t \mid X_1, \ldots, X_{t-1}) \]

\[ P(\text{mat} \mid \text{the cat sat on the}) \approx \frac{\text{count(“the cat sat on the mat”)}}{\text{count(“the cat sat on the”)}} \]
\[ P(X_t | X_1, \ldots, X_{t-1}) \]

How do we estimate these probabilities?

Let’s just count!

\[
P(\text{mat} | \text{the cat sat on the}) = \frac{\text{count(“the cat sat on the mat”)}}{\text{count(“the cat sat on the”)}}
\]

**Challenge:** Increasing \( n \) makes sparsity problems worse.

Typically, we can’t have \( n \) bigger than 5.

Some partial solutions (e.g., smoothing and backoffs) though still an open problem.
Language Models: A History

- Shannon (1950): The predictive difficulty (entropy) of English.

Prediction and Entropy of Printed English

By C. E. Shannon

(Manuscript Received Sept. 15, 1950)

A new method of estimating the entropy and redundancy of a language is described. This method exploits the knowledge of the language statistics possessed by those who speak the language, and depends on experimental results in prediction of the next letter when the preceding text is known. Results of experiments in prediction are given, and some properties of an ideal predictor are developed.
Shannon (1950) build an approximate language model with word co-occurrences.

Markov assumptions: every node in a Bayesian network is conditionally independent of its nondescendants, given its parents.

1st order approximation:

\[ P(\text{mat} \mid \text{the cat sat on the}) \approx P(\text{mat} \mid \text{the}) \]
\( \text{Shannon (1950)} \) build an approximate language model with word co-occurrences.

**Markov assumptions:** every node in a Bayesian network is conditionally independent of its nondescendants, given its parents.

**2\textsuperscript{nd} order approximation:**

\[ P(\text{mat} \mid \text{the cat sat on the}) \approx P(\text{mat} \mid \text{on the}) \]

[Prediction and Entropy of Printed English, Shannon 1950]
Shannon (1950) build an approximate language model with word co-occurrences.

Markov assumptions: every node in a Bayesian network is conditionally independent of its nondescendants, given its parents.

3rd order approximation:

\[ P(mat \mid the \, cat \, sat \, on \, the) \approx P(mat \mid sat \, on \, the) \]
\[ P(X_t \mid X_{1}, \ldots, X_{t-1}) \]

Shannon (1950) build an approximate language model with word co-occurrences.

Then, we can use counts of approximate conditional probability. Using the 3\textsuperscript{rd} order approximation, we can:

\[ P(\text{mat} \mid \text{the cat sat on the}) \approx P(\text{mat} \mid \text{sat on the}) = \frac{\text{count(“sat on the mat”)}}{\text{count(“on the mat”)}} \]

[Prediction and Entropy of Printed English, Shannon 1950]
N-gram Language Models

- **Terminology:** *n*-gram is a chunk of *n* consecutive words:
  - unigrams: “cat”, “mat”, “sat”, ...
  - bigrams: “the cat”, “cat sat”, “sat on”, ...
  - trigrams: “the cat sat”, “cat sat on”, “sat on the”, ...
  - four-grams: “the cat sat on”, “cat sat on the”, “sat on the mat”, ...

- *n*-gram language model:

\[
P(X_t|X_1, \ldots, X_{t-1}) \approx P(X_t|X_{t-n+1}, \ldots, X_{t-1})
\]

[Prediction and Entropy of Printed English, Shannon 1950]
Pre-Computed N-Grams

Google Books Ngram Viewer

Google n-gram viewer https://books.google.com/ngrams/
Data: http://storage.googleapis.com/books/ngrams/books/datasetv2.html
Pre-Computed N-Grams

Google Books Ngram Viewer

Google n-gram viewer https://books.google.com/ngrams/
Data: http://storage.googleapis.com/books/ngrams/books/datasetsv2.html
Language models can tell us something about us ...

Google n-gram viewer https://books.google.com/ngrams/
Data: http://storage.googleapis.com/books/ngrams/books/datasetsv2.html
Pre-Computed N-Grams

Google Books Ngram Viewer

Language models can tell us something about us ...

Google n-gram viewer https://books.google.com/ngrams/
Data: http://storage.googleapis.com/books/ngrams/books/datasetsv2.html