1 Foundations

Language Model
A language model is a distribution over Σ^*, where Σ is a non-empty alphabet.

Globally Normalized Model
For an *energy fct* $\hat{p}:\Sigma^* \to \mathbb{R}$, a GNM is defined as $p(x) = e^{-\hat{p}(x)}/\sum_{y\in\Sigma^*} e^{-\hat{p}(y)}$.

Locally Normalized Language Model
Given the cond. probabilities $p(y \mid y_{<t})$, the corresponding LNLM is

$p_{LN}(y) = p_{SM}(\text{EOS} \mid y)\prod_{t=1}^{|y|} p_{SM}(y_t|y_{<t})$.

Characterizing Tightness
Let

$$p_{\text{EOS}}(t) = \frac{\sum_{\omega\in\Sigma^{t-1}} p_{LN}(\omega)p_{LN}(\text{EOS} \mid \omega)}{\sum_{\omega\in\Sigma^{t-1}} p_{LN}(\omega)}.$$

Then, p_{LN} is **tight** iff $p_{\text{EOS}}(t) = 1$ for some $t \geq 1$ or $\sum_{t=1}^{\infty} p_{\text{EOS}}(t) = \infty$. In particular, if $p_{LN}(\text{EOS} \mid y) \geq f(t)$ for all $y \in \Sigma^t$ and $\sum_{t=1}^{\infty} f(t) = \infty$, then p_{LN} is tight.

Softmax

$$\text{softmax}(x)_i = \frac{\exp(\frac{x_i}{\tau})}{\sum_{j=1}^{n}\exp(\frac{x_j}{\tau})}$$

As $\tau \to \infty$, becomes uniform and as $\tau \to 0$, becomes spiked. We have

$$\text{softmax}(x) = \operatorname*{argmax}_{p\in\Delta^{n-1}} p^\top x - \tau\sum_{i=1}^{n} p_i \log p_i.$$

Representation-based Language Model
An embedding matrix $\mathbf{E}$ and an ecoding fct. $\text{enc}:\Sigma^* \to \mathbb{R}^d$ define a LNLM as

$$p(y_t \mid y_{<t}) = \text{softmax}\left(\mathbf{E}\text{enc}(y_{<t})\right)_{y_t}.$$

Tightness of Softmax RBLMs
If $sz(t) \leq \log t$ for all $t \geq N$ for some N, then the induced model is **tight**. Here, $s = \max_{y\in\Sigma} \|\text{e}(y) - \text{e}(\text{EOS})\|_2$ and $z(t) = \max_{\omega\in\Sigma^t}\|\text{enc}(\omega)\|_2$. In particular, if enc is bounded, then the model is **tight**.

2 Finite State LMs

Finite State Automaton
A FSA is a tuple $\mathcal{A} = (Q, \Sigma, \delta, I, F)$, where Q is a finite set of states, Σ is an alphabet, $\delta \subseteq Q\times(\Sigma\cup\{\epsilon\})\times Q$ are the transitions, and $I, F \subseteq Q$ are the sets of initial/final states.

Weighted FSA
A WFSA is a tuple $\mathcal{A} = (Q, \Sigma, \delta, \lambda, \rho)$, where $\delta \subseteq Q\times(\Sigma\cup\{\epsilon\})\times\mathbb{R}\times Q$ are the weighted transitions and $\lambda, \rho : Q \to \mathbb{R}$ are the initial/final weights.

Probabilistic FSA
A WFSA is probabilistic if λ, ρ and the weights are non-negative, $\sum_{q\in Q}\lambda(q) = 1$ and for all $q \in Q$ we have

$$\rho(q) + \sum_{q\xrightarrow{a/w}q'} w = 1.$$

Path Weights
The weight of a path $\pi = q_1 \xrightarrow{a_1/w_1} q_2 \cdots q_N$ in a WFSA $\mathcal{A}$ is

$$w(\pi) = \lambda(q_1)\prod_{i=1}^{N} w_i \rho(q_N).$$

WFSA Allsum
The allsum of a WFSA $\mathcal{A}$ is

$$Z(\mathcal{A}) = \sum_{y\in\Sigma^*}\mathcal{A}(y) = \sum_{y\in\Sigma^*}\sum_{\pi\in\Pi(A,y)} w(\pi).$$

We have

$$Z(A) = \vec{\lambda}\sum_{d=0}^{\infty} T^d\vec{\rho} = \vec{\lambda}(I-T)^{-1}\vec{\rho}$$

where T is the transition matrix of A.

Tightness of PFSA
A state $q \in Q$ is accessible if there exists a non-zero weighted path from an initial state to q. It is co-accessible if there exists a non-zero weighted path from q to a final state.
A PWFSA is **tight** if and only if all accessible states are co-accessible.

3 Pushdown LMs

Context Free Grammar
A CFG is a tuple $\mathcal{G} = (\Sigma, \mathcal{N}, S, \mathcal{P})$, where Σ is an alphabet of terminals, $\mathcal{N}$ is a set of non-terminals with $\mathcal{N}\cap\Sigma = \emptyset$, $S \in \mathcal{N}$ is the start symbol and $\mathcal{P}$ is a set of production rules of the form $X \to \alpha$ where $X \in \mathcal{N}$ and $\alpha \in (\Sigma\cup\mathcal{N})^*$.

Weighted CFG
A WCFG is a CFG with an associated weight function $\mathcal{W}:\mathcal{P}\to\mathbb{R}$.

Probabilistic CFG
A WCFG is probabilistic if $\mathcal{W}$ is non-negative and for all $X \in \mathcal{N}$ we have $\sum_{X\to\alpha\in\mathcal{P}}\mathcal{W}(X\to\alpha) = 1$.

WCFG Allsum
The allsum of a WCFG $\mathcal{G}$ is

$$\begin{aligned} Z(\mathcal{G}) &= \sum_{d\in\mathcal{D}_\mathcal{G}} w(d) \\ &= \sum_{d\in\mathcal{D}_\mathcal{G}}\prod_{(X\to\alpha)\in d}\mathcal{W}(X\to\alpha), \end{aligned}$$

where $\mathcal{D}_\mathcal{G}$ is the set of all possible derivations in $\mathcal{G}$.

Tightness of PCFG
For a PCFG $\mathcal{G}$ with $|\mathcal{N}| = N$ we define for each $X_n \in \mathcal{N}$ its production generating fct.

$$g_n\left((s_i)_{i=1}^N\right) = \sum_{X_n\to\alpha}\mathcal{W}(X_n\to\alpha)s_1^{r_1(\alpha)}\cdots s_N^{r_N(\alpha)}$$

where $r_i(\alpha)$ is the number of times X_i appears in α. Then we set $E \in \mathbb{R}^{N\times N}$ to have entries $E_{nm} = \frac{\partial}{\partial s_m}g_n(s_1,\ldots,s_N)\Big|_{s_1,\ldots,s_N=1}$.
Then $\mathcal{G}$ is **tight** if $\lambda < 1$ and non-tight if $\lambda > 1$, where $\lambda = \max\left\{|\lambda'| \mid \lambda' \in \sigma(E)\right\}$.

Pushdown Automaton
A language is context-free if and only if it is recognized by some PDA.

Multi-Stack PDA
Any (probabilistic) 2-stack PDA is Turing complete. Hence, the tightness of a probabilistic 2-stack PDA is undecidable.

4 RNNs

RNN
A RNN is given by an initial state $h_0 \in \mathbb{R}^d$ and a dynamics map $h_t = f(h_{t-1}, y_t)$. An RNN-LM uses $\text{enc}(y_{<t+1}) = h_t$.

Elman RNN
An Elman RNN is an RNN with $f(h_{t-1}, y_t) = \sigma(Uh_{t-1} + Ve'(y_t) + b)$, where $U \in \mathbb{R}^{d\times d}$, $V \in \mathbb{R}^{d\times R}$ and $b \in \mathbb{R}^d$ and $e':\Sigma\to\mathbb{R}^R$ is an input embedding function.

Jordan RNN
A Jordan RNN is an RNN with

$$f(h_{t-1}, y_t) = \sigma(U\sigma'(Eh_{t-1}) + Ve'(y_{t-1}) + b).$$

Tightness of RNN-LMs
If the LM uses the softmax and $s\|h_t\| \leq \log t$ (in particular if f is bounded, e.g. if f uses a bounded activation function), then the induced LM is **tight**.

Expressiveness of RNNs
Heaviside Elman RNNs (over $\overline{\mathbb{R}}$) are equivalent to *deterministic* PFSAs. The argument generalizes to any activation function with finite image, in particular any activation implemented on a computer. Minsky's construction encodes any dPFSA using $U \in \mathbb{R}^{|\Sigma||Q|\times|\Sigma||Q|}$ to encode which states are reachable from h_{t-1} and $V \in \mathbb{R}^{|\Sigma||Q|\times|\Sigma|}$ to encode which states can be transitioned to using y_t. It is possible to reduce the hidden state dimensionality to $\Omega(|\Sigma|\sqrt{|Q|})$.
Saturated Sigmoid Elmann RNNs are Turing complete (because they can encode two-stack PDAs). It is thus undecidable whether an RNN-LM is **tight**.

5 Transformers

Soft Attention
Given queries $Q \in \mathbb{R}^{n\times d}$, keys $K \in \mathbb{R}^{t\times d}$ and values $V \in \mathbb{R}^{t\times d}$, soft attention is

$$\text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V.$$

Time/Space compl.: $O(t^2d)$, $O(t^2 + ld)$
Kernelized attention is $O(rtd)$ and $O(rl + rd + ld)$ with feature map dimension r.

Multi-head Attention
Given a context $C \in \mathbb{R}^{t\times d}$ and a query $x \in \mathbb{R}^d$, we set

$$\begin{aligned} \text{MHA}(x) &= \text{Concat}(\text{head}_1,\ldots,\text{head}_N)W_o \\ \text{head}_i &= \text{Att}(xW_q^{(i)}, CW_k^{(i)}, CW_v^{(i)}), \end{aligned}$$

where $W_q^{(i)}, W_k^{(i)}, W_v^{(i)} \in \mathbb{R}^{d\times d_h}$, $W_o \in \mathbb{R}^{d\times d}$ and usually $d_h = d/N$.

Transformer Layer

A transformer layer (without layer-norm) is a function $T: \mathbb{R}^{T\times d} \to \mathbb{R}^{T\times d}$ that maps $\mathbf{X} = (x_1, \dots, x_T)$ to $(z_1, \dots, z_T)$, where

$$a_t = \mathrm{Att}\big(q(x_t), K(\mathbf{X}_t), V(\mathbf{X}_t)\big) + x_t$$
$$z_t = \mathrm{FFN}(a_t) + a_t.$$

Transformer

A transformer is a rep.-based LM with $\mathrm{enc}(y_{<t+1}) = h_t$, where

$$\mathbf{X}_1 = (e'(y_0), \dots, e'(y_t))$$
$$\mathbf{X}_{l+1} = T_l(\mathbf{X}_l)$$
$$h_t = F(x_t^L)$$

for some $F: \mathbb{R}^d \to \mathbb{R}^d$, $e': \Sigma \to \mathbb{R}^d$ and transformer layers $T_1, \dots, T_L$.

Tightness of Transformers

Any transformer using soft attention is **tight** (because its layers are continuous and the set of possible inputs to the first layer is compact, making enc bounded).

Expressiveness of Transformers

Let p_{LN} be an n-gram language model. Then, there exists a transformer $\mathcal{T}$ with $L(p_{LN}) = L(\mathcal{T})$.

6 Sampling

Ancestral Sampling

1. Locally normalize.
2. Sample $y_t \sim p(\cdot \mid y_{<t})$, stop when $y_t = \mathrm{EOS}$.

May not halt $\to$ set max string length.

Sampling Adaptors

To calibrate p we can postprocess probabilities by a function $\alpha: \Delta^{|\Sigma|-1} \to \Delta^{|\Sigma|-1}$.

Top-K Sampling

Set $p(y_t \mid y_{<t}) = 0$ for all but the K most probable tokens, and renormalize.

Nuclues Sampling

Only take top $p\%$ of probability mass.

7 Transfer Learning

ELMo

Assume we have a forward and a backward LM using L LSTM layers. The ELMo representation for a token y_t is

$$\gamma^{\text{task}} \sum_{l=0}^{L} s_l^{\text{task}} h_{tl}^{LM},$$

where $s_l^{\text{task}} \geq 0$, $h_{tl}^{LM} = (\overrightarrow{h}_{tl}^{LM}, \overleftarrow{h}_{tl}^{LM})$, $\overrightarrow{h}_{tl}^{LM}$ and $\overleftarrow{h}_{tl}^{LM}$ are the hidden states of the LM layers.

BERT

BERT is an encoder transformer pre-trained using masked language modelling and next sentence prediction.

Adapters

For $h \in \mathrm{MHA}(C, x), \mathrm{FFN}(x)$, we set $h \leftarrow h + f(hW_1 + b_1)W_2 + b_2$.
$N_{\text{param}} = 2N(2dm + d + m)$

LoRA

Replace weight matrices $W \in \mathbb{R}^{d\times r}$ with $W \leftarrow W + \frac{\beta}{b}AB$ where $A \in \mathbb{R}^{d\times b}$ and $B \in \mathbb{R}^{b\times r}$ are random matrices and β is a constant in b.
$N_{\text{param}} = NH(3b(d + r) + 2bd)$

Prefix Tuning

Prepending each layer with l embedding vectors results in $N_{\text{param}} = Nld$.

8 Parameter Efficient Fine-Tuning

BitFit

Only optimize (a subset of) bias terms.

Diff Pruning

Learn (sparse) δ in $\theta_{\text{FT}} = \theta_{\text{LM}} + \delta$.

Adapters

Insert bottleneck MLPs after each sublayer (MHA and FFN).

9 Knowledge Enhancement

kNN-LM

Store all embedded prefixes and their following words in a database. At inference time, retrieve the k nearest neighbors of a prefix and normalize the exponentiated distances to a probability distribution p_ξ over words. Then sample from a convex combination of p_ξ and the LM.
Dynamic Gating: Set the weighting of distributions depending on the prefix.

10 RLHF

1. Collect a dataset of instructions+answers and train a supervised baseline model.
2. Produce a dataset of comparisons of different answers given by the baseline model, score them manually and train a reward model.
3. Use PPO to fine-tune a LM (the policy) using the reward model.

11 Distributed SGD

Communication Patterns

Centralized, Asynchronous, Decentralized

Logical Channels

Lossless, Sparsification, Quantization

Convergence

C: $O(\frac{1}{\sqrt{nT}})$

CQ: $O(\frac{1}{\sqrt{nT}} + \frac{\epsilon}{\sqrt{T}})$

A: $O(\frac{1}{\sqrt{nT}} + \frac{\tau}{\sqrt{T}})$

D: $O(\frac{1}{\sqrt{nT}} + \frac{\rho}{T^{1.5}})$

12 Data Quality

Importance Measures

LOO, Uniform Weights, Shapley Value

13 Security

Adversarial Examples

Perturb example with noise δ to force misclassification:

$$\begin{aligned} &\text{maximize} && L(f_\theta(x + \delta), y) \\ &\text{subject to} && \|\delta\|_\infty \leq 1\% \end{aligned}$$

Solve with projected Gradient Descent. Doesn't work for text as $x + \delta$ is highly unlikely to be a token embedding. We can instead solve $\arg\max_v (E_v - x_i)^\top \nabla_{x_i} L$ and replace x_i by v.

14 Privacy

Data Secrecy

Central server sees all training data.
Gold Standard Solutions: Secure multi-party computation, fully homomorphic encryption $\rightsquigarrow$ slow & expensive.
Federated Learning. Clients send gradients. Can be attacked with optimization.
Weight-trap attack: Server sends model s.t. $\nabla_\theta L(f_\theta(x_i)) = x_i$.

Data Memorization

Can generate lots of text and filter text where model is abnormally confident.
Defenses:
1. Filter memorized outputs. Problem: Exact matches are not enough.
2. Deduplicate training data.

Differential Privacy

An algorithm M is ε-differentially private if for any "neighboring" databases D_1, D_2 that differ in a single element, and any output S we have:

$$P\big[M(D_1) \in S\big] \leq \exp(\varepsilon) P\big[M(D_2) \in S\big].$$

Post-Processing: If M is ε-DP, then $f(M)$ for any function f is also ε-DP.
Composition: If M_1 is ε_1-DP and M_2 is ε_2-DP then $f(M_1, M_2)$ is $(\varepsilon_1 + \varepsilon_2)$-DP.