1. Probabilistic Modeling:

1.1 LMs: Distr. over Strings

Def: An Alphabet Σ/Δ is a finite non-empty set. Elements $a, b, c \in \Sigma$ are called symbols/letters. $y_n \in \Sigma$

Def: A string over Σ is any finite seq of letters: $y = y_1 \cdots y_T$

Def: The Kleene star Σ^* is def. as: $\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$ where $\Sigma^n := \underbrace{\Sigma \times \dots \times \Sigma}_{n \text{ times}}$ & $\Sigma^0 = \{\varepsilon\}$, ε = empty string.

We call Σ^* the Kleene closure of Σ. Def: $\Sigma^+ = \bigcup_{n=1}^{\infty} \Sigma^n = \Sigma\Sigma^*$

Def: A language L is a subset of Σ^*

Def: A LM is a (discrete) distr. p_{LM} over Σ^*

Def: The weighted lang. of p_{LM} is def as $L(p_{LM}) = \{(y, p_{LM}(y)) \mid y \in \Sigma^*\}$

2.2 Global & Local Normalization

Def: Let $\hat{p}_{GN}(y): \Sigma^* \to \mathbb{R}$ be an energy fct. A globally norm. model (GNM) is defined as:

$$p_{LM}(y) := \frac{\exp(-\hat{p}_{GN}(y)/T)}{\sum_{y' \in \Sigma^*} \exp(-\hat{p}_{GN}(y')/T)} =: \frac{\exp(-\hat{p}_{GN}(y)/T)}{Z_G}$$

T: temp param.

Rem: ⊕: only depends on $\hat{p}_{GN}$, scores entire seq at once prob distr. over Σ^* directly ⊖: Z_G difficult to compute

Thm: Any normalizable E-fct ($Z_G < \infty$) induces a LM over Σ^*.

Proof: $\sum_{y \in \Sigma^*} p_{GN}(y) = \dots = 1$

⊕LNLM/⊖GNLM: GNM difficult to normalize, difficult to sample from (entire string has to be gen at once. → sol: LNLM: symbolwise

Def: $\bar{\Sigma} = \Sigma \cup \{EOS\}$, $y \in \bar{\Sigma}^*$, $\bar{y} \in \bar{\Sigma}$

Def: A seq model (SM) over Σ is def as a set of cond. prob. distr. $p_{SM}(y \mid \boldsymbol{y})$ for $y \in \Sigma$, $\boldsymbol{y} \in \Sigma^*$.

Def: Let p_{SM} be a SM over $\bar{\Sigma}$. A locally normalized LM (LNM) over Σ is def as: $p_{LN}(y) := p_{SM}(EOS \mid y) \prod_{t=1}^{T} p_{SM}(y_t \mid y_{<t})$ for $y \in \Sigma^*$

A LNM is tight if $\sum_{y \in \Sigma^*} p_{LN}(y) = 1$ ⟺ $\mathbb{P}(\Sigma^\infty) = 0$

Rem: - SMs (⇒LNM) may place pos. prob on ∞-long seqs.
- LNM model a series of cond. prob. distr.s
- SMs (LNMs) may not be LMs (∞-coin toss model)

Def: Prefix prob: $\pi(y) := \sum_{y' \in \Sigma^*} p_{LM}(yy')$ = prob that y is a prefix of any string yy'.

Thm: Let p_{LM} be LM. Then ∃ a locally normalized LM p_{LN} s.t. $\forall y \in \Sigma^*$ w/ $|y| = T$: $p_{LM}(y) = p_{LN}(y)$ ⟺ every LM can be loc. norm.

Proof: Def: $p_{SM}(y \mid \boldsymbol{y}) = \frac{\pi(\boldsymbol{y}y)}{\pi(\boldsymbol{y})}$, $p_{SM}(EOS \mid y) := \frac{p_{LM}(y)}{\pi(y)}$ def over $\bar{\Sigma}$

⇒ p_{SM} valid SM

Case 1: $\pi(y) > 0$: $p_{LN}(y) = p_{SM}(EOS \mid y) \cdot \prod_{t=1}^{T} p_{SM}(y_t \mid y_{<t}) = \frac{p_{LM}(y)}{\pi(y)} \cdot \frac{\pi(y)}{\pi(y_{<T})} \cdot \frac{\pi(y_{<T})}{\pi(y_{<T-1})} \cdots \frac{\pi(y_1)}{\pi(\varepsilon)} = p_{LM}(y)$

Case 2: $\pi(y) = 0$: There must $\exists\, 1 \le t' \le T$: $\pi(y_{\le t'}) = 0$

↳ $p_{LN}(y) = \prod_{t=1}^{T} p_{SM}(y_t \mid y_{<t}) = 0$

Rem: Any LM p_{LM} can be converted into a locally normal. seq model. BUT: converse is NOT true.

2.3 Tightness of LMs

Def: A seq model is a probability space over $\Sigma^* \cup \Sigma^\infty$

ReDef: A LM is a prob space over Σ^* ⟺ LM is a SM s.t. $\mathbb{P}(\Sigma^\infty) = 0$

Remarks:

A LNLM is tight iff ∃t s.t. $p(EOS \mid t) = 1$ or $\sum_{t=1}^{\infty} p_{EOS}(t) = \infty$

$\max_X \prod_{i=1}^{n} \mathbb{P}(t_i \mid \text{inst, Browse evil.com}, X, t_1, t_2, \dots, t_{i-1})$ where $t_1, \dots, t_n$ are tokens of the desired output.

Algo (LogitDelta) { lower ← -200; upper ← 200; while upper-lower > eps { mid ← (lower+upper)/2; logit_bias ← {token: mid}; t ← argmax_{t∈Vocab} (f(prompt) + logit_bias[t]); if t == token: upper ← mid; else: lower ← mid } return (lower+upper)/2 } ⇒ nbr of queries: $\mathcal{O}(\log(1/\varepsilon))$

Algo (RedList) { id_red_list = gen_red_list(input_ids[:,-1], vocabsize, frac) for t in id_red_list: { score[:,t] = score[:,t] - self.logit_offset } return scores. }

Algo (Hidden dim extraction): { Init n (s.t. n≫h); Init $Q \in \mathbb{R}^{n \times l}$; for i=1:n { p_i ← rand Prefix(); $Q_i = \mathcal{O}(p_i)$ ($\mathcal{O}$ is oracle) } Calculate singular values (Q): $\lambda_1 \ge \lambda_2 \ge \dots \ge \lambda_n$ Count nonzero λ_i ⇒ return count }

Embed: Attend to a specific word/token: $W^Q = 1 \cdot 1^T = (\vdots)$, $W_K = e^T \cdot 1^T = (e^T, \dots, e^T)$, where e = embed of word/token.

⇒ $U_{ij} = Q(x_i) K(x_j)^T = D(\sum_k x_i^k) e \cdot x_j^T \propto \langle e, x_j \rangle \hat{=}$ cos-sim

Other exmpls: PCs: $W^Q = W^K = (u_0^{cont}, u_3^{cont})$, cos/sin: $W^Q = W^K = (u_0^{cos}, u_0^{sin}, \vdots)$

Att to prev token: $(u_0^{cos}, u_0^{sin}) \cdot x_n^\alpha = \lambda_0(\cos(\alpha_0 n), \sin(\alpha_0 n))$

↳ $W^K = \begin{pmatrix} u_0^{cos} \\ u_0^{sin} \end{pmatrix}$, $W^Q = \begin{pmatrix} R(-\alpha_0) & 0 \\ 0 & R(-\alpha_0) \end{pmatrix} \begin{pmatrix} u_0^{cos} \\ u_0^{sin} \end{pmatrix}$, $R(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$

Def: A LNLM is a LM def through a set of cond prob distr $p_{SM}(\bar{y} \mid y)$ for $\bar{y} \in \bar{\Sigma}$, $y \in \Sigma^*$ ($p_{LN} = \dots \to$)

To be a true LM, it has to be tight.

Def (σ-algebra). $\mathcal{F} \subseteq \mathcal{P}(\Omega)$ is a σ-alg. if: 1) $\Omega \in \mathcal{F}$, 2) $A \in \mathcal{F} \Rightarrow A^c \in \mathcal{F}$ 3) $A_1, A_2, \dots$ (in)finite seq) ⇒ $\bigcup_{n=1}^{\infty} A_n \in \mathcal{F}$

∞ coin toss: $1 = \mathbb{P}(\Sigma^\infty) = \mathbb{P}(\bigcup_{w \in \Sigma^\infty} \{w\}) = \sum_{w \in \Sigma^\infty} \mathbb{P}(\{w\}) = \sum 0 = 0$

Parse Trees: $\mathbb{P}(T) = \prod_{r \in T} p(r)$, String len: $\sum_{y \in \Sigma^*} |y| = \sum_{n=0}^{\infty} n \cdot |\Sigma|^n$

Subregular: A language is subreg if it can be recognized by a FSA or any weaker machine. n-gram LMs are subreg

Strictly local LM: A language L is strictly n-local if $\forall y$ w/ $|y| = n-1$ & $\forall x_1, x_2, z_1, z_2 \in \Sigma^*$: $x_1 y z_1 \in L \wedge x_2 y z_2 \in L \Rightarrow x_1 y z_2 \in L \wedge x_2 y z_1 \in L$. A lang. L is strictly local if it is strictly n-local $\exists n \in \mathbb{N}$.

Rem: Sol of sparsemax often lies on the boundary of prob simplex → sparse sol.

$sparsemax(x) = \arg\min_{p \in \Delta^{D-1}} \|p - x\|_2^2 \to \frac{\partial\, smax(x)_i}{\partial x_j} = \begin{cases} \delta_{ij} - \frac{1}{|S(x)|} & i,j \in S \\ 0 & \text{else} \end{cases}$ w/ $S(x) = supp(smax(x))$

2. Modeling Foundations

2.1 Representation-based LMs

In Rep-based LMs, we param a LNM through SMs $p_{SM}(y \mid \boldsymbol{y})$

Def: A context enc. fct. $enc(.): \bar{\Sigma}^* \to \mathbb{R}^D$ is a repr. fct of {Symbols / Strings} $\bar{y} \in \bar{\Sigma}^*$. Ref to $enc(\bar{y})$ as enc. of $\bar{y}$

Def: The encoding matrix $E \in \mathbb{R}^{|S| \times D}$ is def. as the row-wise embedding of symbols $\bar{y} \in S$ where S is a finit vocabul. D is the embedding dimension.

Prop: Each entry of the vec. $E \cdot enc(\bar{y}) \in \mathbb{R}^{|\bar{\Sigma}|}$ corresp. to the similarities of a particular symbol $\bar{y}$ to the context $\bar{\boldsymbol{y}}$.

↳ Transform $E \cdot enc(\bar{y})$ into prob distr by using proj. fct.

Def: Prob simplex $\Delta^{D-1} := \{x \in \mathbb{R}^D \mid x_d \ge 0, d = 1, \dots, D \ \&\ \sum_{d=1}^{D} x_d = 1\}$

Def: Proj. fct $f_{\Delta^{D-1}}: \mathbb{R}^D \to \Delta^{D-1}$

Def: prob distr: $p_{SM}(\bar{y} \mid \bar{\boldsymbol{y}}) = f_{\Delta^{|\bar{\Sigma}|-1}}(E \cdot enc(\bar{\boldsymbol{y}}))_{\bar{y}}$

Def: Softmax: $\tau \in \mathbb{R}_+$. $softmax(x)_d := \exp(\frac{x_d}{\tau}) / \sum_{j=1}^{D} \exp(\frac{x_j}{\tau})$

Thm: τ high → $\mathcal{U}$-distr, τ small → peaky distr.

↳ $\lim_{\tau \to \infty} softmax(x) = \frac{1}{D}\mathbf{1}$, $\lim_{\tau \to 0^+} softmax(x) = e_{\arg\max(x)}$

Thm: $softmax(x + c \cdot \mathbf{1}) = softmax(x)$ $\forall c \in \mathbb{R}$, $x_i \le x_j \Rightarrow sm(x)_i \le sm(x)_j$,

$\frac{\partial}{\partial x_j} softmax(x)_i = \frac{\delta_{ij} \exp(x_i) \cdot \sum_k \exp(x_k) - \exp(x_i)\exp(x_j)}{(\sum_k \exp(x_k))^2}$ $\{= \mathbb{1}\{i=j\} sm(x)_i - sm(x)_i \cdot sm(x)_j\}$

Def: A representation-based locally normalized model is a model of the form: $p_{SM}(\bar{y}_t \mid \bar{y}_{<t}) = f_{\Delta^{|\bar{\Sigma}|-1}}(E \cdot enc(\bar{y}_{<t}))_{\bar{y}_t}$

Normally: $f_{\Delta^{|\bar{\Sigma}|-1}}$ = softmax. Probab. of an entire string $y \in \Sigma^*$: $p_{LN}(y) := p_{SM}(EOS \mid y) \cdot \prod_{t=1}^{T} p_{SM}(y_t \mid y_{<t})$ with y_0 = BOS

Prop: If $p_{LN}(EOS \mid y) \ge f(t)$ $\forall y \in \Sigma^t$ $\forall t$ ∧ $\sum_{t=1}^{\infty} f(t) = \infty$ ⇒ p_{LN} tight

Thm: Let $s := \sup_{y \in \Sigma} \|e(y) - e(EOS)\|_2$ & $z_{max} := \max_{y \in \Sigma^t} \|enc(y)\|_2$. The LNM p_{LN} induced by p_{SM} is tight if $s \cdot z_{max} \le \log(t)$

Thm: A LN Rep-based LM w/ $\mathcal{U}$-bounded $\|enc(y)\|_p$ is tight.

3. Finite State LMs

In FS LMs there are only finitely many contexts $\boldsymbol{y}$ which define the distr. over the next symbol, $p_{LM}(y \mid \boldsymbol{y})$.

3.1 Weighted FSA

Def: A FSA is a 5-tuple $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ where: Σ is an alphabet, Q is a set of states (finite), $I \subseteq Q$ the set of init states, $F \subseteq Q$ the set of final/accepting states, $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$. Elements of δ called transitions.

Def: A FSA $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ is deterministic if:
(i) it does not have any ε-transitions ($q \xrightarrow{\varepsilon} q'$)
(ii) $\forall (q, a) \in Q \times \Sigma$ ∃ at most one $q' \in Q$ s.t. $q \xrightarrow{a} q' \in \delta$
(iii) there is a single init state, i.e. $|I| = 1$

Otherwise, $\mathcal{A}$ is non-deterministic.

Def: Let $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ be a FSA. The language of $\mathcal{A}$: $\mathcal{L}(\mathcal{A}) = \{y \mid y \text{ is recognized by } \mathcal{A}\}$

Def: A language $L \subseteq \Sigma^*$ is regular, iff it can be recognized by an unweighted FSA ⟺ $\exists \mathcal{A}: L = \mathcal{L}(\mathcal{A})$

Def: A real-weighted FSA (WFSA): $\mathcal{A} = (\Sigma, Q, \delta, \lambda, \rho)$: Σ is a finite alphabet, Q is a finite set of states, $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \mathbb{R} \times Q$ a finite set of transitions, ($q \xrightarrow{a/w} q'$), $\lambda: Q \to \mathbb{R}$ & $\rho: Q \to \mathbb{R}$ weighting fct's.

Rem: $I = \{q \in Q \mid \lambda(q) \ne 0\}$, $F = \{q \in Q \mid \rho(q) \ne 0\}$, $w := \omega(q \xrightarrow{a/w} q')$

Def: Let $\mathcal{A} = (\Sigma, Q, \delta, \lambda, \rho)$ be a WFSA. $\forall a \in \Sigma$, we def the symbol-specific transition matrix $T^{(a)}$ as the trans. mat. of the graph restricted to a-labeled transitions. Def the full trans. mat. as $T := \sum_{a \in \Sigma} T^{(a)}$ (row ⟺ q from & col ⟺ q' to)

Def: A path $\pi \in \delta^*$ is a collection of consequtive transitions. Denote sets of paths w/ Π. E.g.: $\Pi(\mathcal{A})$: set of all paths in $\mathcal{A}$, $\Pi(\mathcal{A}, y)$: set of all paths in $\mathcal{A}$ that yield $y \in \Sigma^*$, $\Pi(\mathcal{A}, q, q')$: set of all paths in $\mathcal{A}$ from q to q'. $q_1 \xrightarrow{a_1/w_1} q_2 \cdots q_{N-1} \xrightarrow{a_N/w_N} q_N$

Def: The inner path weight $w_I(\pi)$ of path π ($|\pi| = N$) is def.: $w_I(\pi) = \prod_{n=1}^{N} w_n$. The (full) path weight is def as: $w(\pi) = \lambda(p(\pi))\, w_I(\pi)\, \rho(n(\pi))$ where: {p(π): origin of π; n(π): destination}

A path π is "accepting"/"successful" if $w(\pi) \ne 0$.

Def: The stringsum/string weight/acceptance weight of a string $y \in \Sigma^*$ under a WFSA $\mathcal{A}$ is: $\mathcal{A}(y) := \sum_{\pi \in \Pi(\mathcal{A}, y)} w(\pi)$

Def: Let $\mathcal{A}$ be a WFSA. It's (weighted) language is: $\mathcal{L}(\mathcal{A}) := \{(y, \mathcal{A}(y)) \mid y \in \Sigma^*\}$

Def: A weighted regular language L s.t. ∃ WFSA $\mathcal{A}$: $L = \mathcal{L}(\mathcal{A})$

Def: Let $\mathcal{A}$ WFSA. The allsum of $q \in Q$ is def: $Z(\mathcal{A}, q) = \sum_{\pi \in \Pi(\mathcal{A}), q_1 = q} w_I(\pi)\, \rho(n(\pi))$

Def: $\mathcal{A}$ WFSA. The allsum of $\mathcal{A}$: $Z(\mathcal{A}) = \sum_{y \in \Sigma^*} \mathcal{A}(y) = \sum_{\pi \in \Pi(\mathcal{A})} w(\pi)$

Def: A WFSA $\mathcal{A}$ is normalizable if $Z(\mathcal{A}) < \infty$

Def: A WFSA $\mathcal{A} = (\Sigma, Q, \delta, \lambda, \rho)$ is probabilistic PFSA if $\sum_{q \in Q} \lambda(q) = 1$ & $\forall q \in Q \wedge \forall q \xrightarrow{a/w} q' \in \delta$ it holds: $\lambda(q) \ge 0$, $\rho(q) \ge 0$, $w \ge 0$ & $\sum_{q \xrightarrow{a/w} q'} w + \rho(q) = 1$

Rem: init weight of all states form a prob distr

3.2 Finite-state LMs:

Def: A LM p_{LM} is finite-state if ∃ WFSA $\mathcal{A} = (\Sigma, Q, \delta, \lambda, \rho)$ s.t. $\mathcal{L}(\mathcal{A}) = L(p_{LM})$. ⟺ p_{LM} is a weighted regular lang.

Def prob := weight of π, Stringsum = prob of y: $p_{\mathcal{A}}(y) = \mathcal{A}(y)$

Def: Let $\mathcal{A} = (\Sigma, Q, \delta, \lambda, \rho)$ be a normalizable WFSA with non-neg weights. We define the prob of a string $y \in \Sigma^*$ under $\mathcal{A}$ as: $p_{\mathcal{A}}(y) := \mathcal{A}(y) / Z(\mathcal{A})$

Def: Let $\mathcal{A} = (\Sigma, Q, \delta, \lambda, \rho)$ be a WFSA. We define the LM induced by $\mathcal{A}$ as the following prob distr over Σ^*: $p_{LM_{\mathcal{A}}}(y) = p_{\mathcal{A}}(y)$

3.3 Normalizing Finite-State LMs

Rem: Let T be a trans. mat: $T_{ij} = \sum_{\pi \in \Pi(\mathcal{A}, i, j)} w_I(\pi)$. Then:

$Z(\mathcal{A}) = \sum_{\pi \in \Pi(\mathcal{A})} w(\pi) = \sum_{i,j \in Q} \sum_{\pi \in \Pi(\mathcal{A}, i, j)} \lambda(i)\, w_I(\pi)\, \rho(j) = \vec{\lambda} M \vec{\rho}$

Lemma: Let T be the trans. mat. of some weighted directional graph G. Then the matrix T^d contains the allsum of all paths of length exactly d, i.e.: $T^d_{ij} = \sum_{\pi \in \Pi(\mathcal{A}, i, j), |\pi| = d} w_I(\pi)$ ⇒ $T^{\le d} = \sum_{k=1}^{d} T^k$

Def: $T^* = \lim_{d \to \infty} T^{\le d} = \sum_{d=0}^{\infty} T^d$

↳ $T^* = \mathbf{1} + T \cdot T^*$ ⇒ $T^* = (\mathbf{1} - T)^{-1}$ → Mat inv. ~ $\mathcal{O}(|Q|^3)$

Rem: If $\|T\|_2 < 1$ (All EV of T have norm smaller 1) then $1 - T$ is invertible & T^* exists

Thm: Normalizable WFSA with non-neg weights & a tight PFSA are equally expressive

Lemma: Let $\mathcal{A} = (\Sigma, Q, \delta, \lambda, \rho)$ & $q \in Q$. Then:

$Z(\mathcal{A}, q) = \sum_{q \xrightarrow{a/w} q' \in \delta_{\mathcal{A}}} \omega(q \xrightarrow{a/w} q')\, Z(\mathcal{A}, q') + \rho(q)$

Rem: Hints for Proof of thm: $\lambda_{\mathcal{A}_L}(q) = \lambda(q) \frac{Z(\mathcal{A}, q)}{Z(\mathcal{A})}$, $\rho_{\mathcal{A}_L}(q) = \rho(q) / Z(\mathcal{A}, q)$, $\omega_{\mathcal{A}_L}(q \xrightarrow{a/w} q') = \omega(q \xrightarrow{a/w} q') \frac{Z(\mathcal{A}, q')}{Z(\mathcal{A}, q)}$, $\sum_{q \xrightarrow{a/w} q' \in \delta_{\mathcal{A}_L}} w + \rho_{\mathcal{A}_L}(q) = \dots = 1$, $\sum_{q \in Q} \lambda_{\mathcal{A}_L}(q) = \dots = 1$, $p_{\mathcal{A}}(\pi) = \frac{w(\pi)}{Z(\mathcal{A})} = p_{\mathcal{A}_L}$

3.4 Tightness of Finite-State Models:

Rem: Any normalizable globally norm. FS LM is tight by def. Can focus on tightness of LN FS models. LN FS models are exactly PFSAs.

Thm: A PFSA is tight iff all accessible states are also co-accessible. (If $\lambda(q) \ne 0 \to \exists\, q \xrightarrow{a/w} q'$ s.t. $w > 0$ & $\rho(q') \ne 0$)

3.5 n-gram Models:

Rem: #params n-gram: $\mathcal{O}(|\Sigma|^n)$ $= (|\bar{\Sigma}| - 1) \cdot \sum_{j=0}^{n-1} |\Sigma|^j$

Def: In an n-gram model the cond prob of $y_t \mid y_{<t}$ only depends on n-1 prev symb. $y_{t-n+1}^{t-1} := y_{t-1} \cdots y_{t-n+1}$: $p_{SM}(y_t \mid y_{<t}) = p_{SM}(y_t \mid y_{t-n+1}^{t-1})$

Rem: By limiting the length of context, the number of possible prob distr. that need to be tracked is limited to $\mathcal{O}(|\Sigma|^{n-1})$

Rem: ⊖: #params, Poor perform. w/ limited data, lack of long range dependencies, prone to getting stuck in loops, lack of semantic understan.

3.6 Representation-based n-gram Models:

To infuse a "similarity"-notion in context, work with embedd. instead of pure text! Let $e(y)$ be embed. of y. → E embed mat

Def: $enc(y_{<t}) := enc(y_{t-1}, y_{t-2}, \dots, y_{t-n+1}) := b + Wx + U \cdot \tanh(d + Hx)$ FFNN, $x = concat(e(y_t), \dots, e(y_{t-n+1}))$

↳ $p_{SM}(\bar{y}_t \mid \bar{y}_{<t}) := softmax(enc(\bar{y}_{<t})^T E + b)_{\bar{y}_t}$

↳ $p_{LN}(y) = softmax(enc(\bar{y}_T, \dots, \bar{y}_{T-n+2})^T E + b)_{EOS} \prod_{t=1}^{T} p_{SM}(y_t \mid y_{<t})$

Rem: Problem of limited context persists. However, it solves issue of 'similarity-knowledge' by consid words similarities & sharing parameters across diff. contexts in the form of encoding fct instead of lookup tables.

Prop: MLE sol: $p_{SM}(y_n \mid y_{<n}) = \theta_{y \mid \boldsymbol{y}} = \frac{\#\{(y_1, \dots, y_n)\}}{\#\{(y_1, \dots, y_{n-1})\}} = \frac{\#\{y_{<n} y_n\}}{\#\{y_{<n}\}}$

Rem: ⊕: model able to take similarity into account, encoding enables parameter sharing across contexts, less params, #params scales with n.

Finite state LMs cannot capture any non-regular (context-free) phenomena. (Expressivity of FSLMs)

4. Pushdown LMs

Rem: Human Lang. is not finite state (recursive hierarchical structure) n-gram models recogn. strictly local languages → n-gram: subregular

Rem: Every n-gram model is a PFSA

↳ n-gram models are finite state (finite history ⇒ finite state)

Rem: n-gram Model = WFSA $\mathcal{A}$ w/: $\Sigma_{\mathcal{A}} := \Sigma$,

$Q_{\mathcal{A}} := \bigcup_{t=0}^{n-1} \{BOS\} \times \Sigma^t$

$\delta_{\mathcal{A}} := \{ y_{t-n:t-1} \xrightarrow{y_t / p_{SM}(y_t \mid y_{t-n:t-1})} y_{t-n+1:t} \mid y_{t-n:t-1} \in \bigcup_{t=0}^{n-2} \{BOS\}^{n-2-t} \times \Sigma^t;\ y_{t-n}, y_t \in \Sigma \}$

$\lambda_{\mathcal{A}}: y \mapsto \mathbb{1}\{y = \underbrace{BOS \dots BOS}_{(n-1)\times}\}$

$\rho_{\mathcal{A}}: y \mapsto p_{SM}(EOS \mid y)$, $y \in Q_{\mathcal{A}}$

$KL(P \| Q) = \sum_{x \in \mathcal{X}} p(x) \log\left(\frac{p(x)}{q(x)}\right)$

Entropy: $H(P) = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$

⇒ $KL(P, Q) = H(P, Q) - H(P)$

Def: $rel_t = \mathbb{1}\{\text{elem@}t \text{ is relev.}\}$, Prec@k $= \frac{TP}{TP + FP} = \frac{TP}{k}$ = how many items within top k relevant

Avr. prec. @k $= \frac{1}{\#positives} \sum_{t=1}^{K} p@t \cdot rel_t$ — Metrics for RAG

Transformer:

Encoder Block: (multihead) self-att. → LN → Fully Con → LN (skip con.)

Decoder Block: (MH) masked self att → LN → (MH) Cross-Att → LN → Fully Connected → LN

Encoder: Bidir. self att mech over entire seq

Decoder: Autoregr. Gen. Masked self att. + Cross Att: retrieve info when gen. output

Weights: $W^Q, W^K, W^V \in \mathbb{R}^{D/H \times D}$, $W^O \in \mathbb{R}^{D \times D}$

Interpret: Q&K are used to compute the pairwise relevance/attention score btw. two tokens

↳ Q: What you want, K: Keywords of the values, V: Values you want to look up

$X \in \mathbb{R}^{T \times D}$, $V_t = XW^V \in \mathbb{R}^{T \times D}$

5. Neural Network LMs:

5.1 RNNs:

Def: Let Σ be an alphabet. A (deterministic) real valued RNN $\mathcal{R}$ is a 4-tuple (Σ, D, f, h_0) where: Σ alphabet of input symbols; D dim of $\mathcal{R}$; $f: \mathbb{R}^D \times \Sigma \to \mathbb{R}^D$ dynamics map; $h_0 \in \mathbb{R}^D$ is the initial state.

Def: Let $\mathcal{R}$ be a RNN. A recurrent neural encoding fct $enc_{\mathcal{R}}$ is a repr. fct: $enc_{\mathcal{R}}(y_{\le t}) := f(enc_{\mathcal{R}}(y_{<t}), y_t) \in \mathbb{R}^D$. Also called hidden state: $h_t := enc_{\mathcal{R}}(y_{\le t}) = f(h_{t-1}, y_t)$

Def: Let $\mathcal{R}$ be RNN & $E \in \mathbb{R}^{|\bar{\Sigma}| \times D}$ a symbol repr. mat. A D-dim recurrent neural seq model over alph. Σ is a tuple (Σ, D, f, E, h_0) def the seq model: $p_{SM}(y_t \mid y_{<t}) := f_{\Delta^{|\bar{\Sigma}|-1}}(E \cdot enc_{\mathcal{R}}(y_{<t})) = f_{\Delta^{|\bar{\Sigma}|-1}}(E h_{t-1})_{y_t}$ often: $= softmax(E \cdot h_{t-1})_{y_t}$

Thm: A softmax RNN seq. Model is tight if $\forall t$: $s \cdot \|h_t\|_2 \le \log(t)$ where $s := \max_{y \in \Sigma} \|e(y) - e(EOS)\|_2$ (⇐ if $|f(x)_d| \le M$ $\forall d, x$)

↳ All standard sigmoid & tanh activ. RNNs are tight.

Def: An Elman seq model $\mathcal{R} = (\Sigma, D, U, V, E, b_h, h_0)$ is a D-dim rec. neur. seq. mod. over alphabet Σ w/ $h_t = \sigma(U h_{t-1} + V e'(y_t) + b_h)$ where $e': \Sigma \to \mathbb{R}^R$ is the input symbol embedding fct, σ: elementwise non-linearity. $b_h \in \mathbb{R}^D$, $U \in \mathbb{R}^{D \times D}$, $V \in \mathbb{R}^{D \times R}$ → #params $= 2DR + D^2 + D$ | $\|U\| < 1$ ⇒ van; σ = ReLU ⇒ expl

Def: A Jordan seq. mod. is a D-dim RN seq mod. over Σ w/ $h_t = \sigma(U r_{t-1} + V e'(y_t) + b_h)$, $r_t = \sigma_o(E h_t)$, params as before

Corollary: Elman & Jordan RNNs w/ bounded activation fct σ & the softmax proj. fct. are tight

Rem: Gating acts as an on/off switch & helps w/ vanishing & exploding gradients. In an ERNN w/ $h \in \mathbb{R}^D$, 2^D possible config.

Def: A LSTM Unit is a RNN w/ the following dynamics map:

$i_t = \sigma(U^i h_{t-1} + V^i e'(y_t) + b^i)$, $f_t = \sigma(U^f h_{t-1} + V^f e'(y_t) + b^f)$

$o_t = \sigma(U^o h_{t-1} + V^o e'(y_t) + b^o)$, $g_t = \tanh(U^g h_{t-1} + V^g e'(y_t) + b^g)$

$c_t = f_t \odot c_{t-1} + i_t \odot g_t$, $h_t = o_t \odot \tanh(c_t)$ (hidden state)

input, forget, output gates, g_t: candidate vec, c_t: memory cell, σ: sigmoid

Def: A gated recurrent unit (GRU) def. a dyn. map. s.t.:

$r_t = \sigma(U^r h_{t-1} + V^r e'(y_t) + b^r)$, $z_t = \sigma(U^z h_{t-1} + V^z e'(y_t) + b^z)$

$g_t = \tanh(U^g(r_t \odot h_{t-1}) + V^g e'(y_t) + b^g)$, $h_t = (1 - z_t) \odot g_t + z_t \odot h_{t-1}$

reset gate, z_t: update gate, g_t: candidate vector.

Rem: Time complex. of RNNs: $\mathcal{O}(T)$ for seq of length T. Parallelization is not possible.

Def: Heaviside fct.: $H(x) = \mathbb{1}\{x > 0\} = \begin{cases} 0 & x \le 0 \\ 1 & x > 0 \end{cases}$

Def: A Heaviside Elman network (HRNN) is an elman network with heaviside fct H as the non-linearity.

Thm: Heaviside Elman RNNs are equiv to deterministic PFSA.

Lemma: The distr. represented by a RNN w/ Heaviside non-lin. is regular.

Proof Thm: Let $\mathcal{R} = (\Sigma, D, U, V, E, b_h, h_0)$ be a HRNN def p_{SM}. We construct a det. PFSA $\mathcal{A} = (\Sigma, Q, \delta, \lambda, \rho)$ def same str.p. $s: \{0,1\}^D \to \{0, \dots, 2^D - 1\}$ bijection, $q := s(h) \in Q = \{0,1\}^D$. def $q \xrightarrow{a/w} q'$ s.t. $q' = s(\sigma(Uh + V[\![y]\!] + b_h))$, $w := p_{SM}(y \mid h) = f_{\Delta^{|\bar{\Sigma}|-1}}(E \cdot h)_y$.
$\lambda(s(h)) = \mathbb{1}\{h = h_0\}$, $\rho(q) := p_{SM}(EOS \mid s(q))$

$\mathcal{A}(y) = \lambda(s(h_0)) \cdot [\prod_{t=1}^{T} w_t] \cdot \rho(q_{T+1}) = 1 \cdot \prod_{t=1}^{T} p_{SM}(y_t \mid s^{-1}(q_t)) \cdot p_{SM}(EOS \mid s^{-1}(q_{T+1})) = p_{LN}(y)$ □

Lemma: Let $\mathcal{A} = (\Sigma, Q, \delta, \lambda, \rho)$ be a tight deterministic PFSA. Then ∃ a HRNN w/ hidden state size $|\Sigma| \cdot |Q|$ that encodes the same distr as $\mathcal{A}$.

↳ Minsky Construct: $h_t := [\![(q_t, y_t)]\!] \in \{0,1\}^{|Q| \cdot |\Sigma|}$, $b_h := -\mathbf{1} \in \mathbb{R}^{|Q||\Sigma|}$

$U \in \mathbb{R}^{|\Sigma||Q| \times |\Sigma| \cdot |Q|}$ s.t.: $U_{n(q', y'), n(q, y)} := \mathbb{1}\{q_t \xrightarrow{y'/\cdot} q' \in \delta\}$

$V \in \mathbb{R}^{|\Sigma||Q| \times |\Sigma|}$ s.t.: $V_{n(q', y'), m(y')} := \mathbb{1}\{\cdot \xrightarrow{y'/\cdot} q' \in \delta\}$

Rem: $U[:, n(q, y)]$ repr. "out-neighborhood" of state q ↳ has 1's at indices corr. to (q', y') s.t. $q \xrightarrow{y'/\cdot} q' \in \delta$. 1's in y'-col in V corr. to (q', y') s.t. q' is reachable with y' from some state. ↳ HRNN only outputs 1 if $\{q_T \xrightarrow{y'/\cdot} q', \cdot \xrightarrow{y'/\cdot} q' \in \delta \Leftrightarrow q_T \xrightarrow{y'/\cdot} q' \in \delta\}$

Constr. of trans. probs:

For sparsemax: $E_{m(y'), n(q, y)} = \begin{cases} \omega(q \xrightarrow{y'/\cdot} \cdot) & \text{if } y' \in \Sigma \\ \rho(q) & \text{if } y' = EOS \end{cases}$

for softmax: $E_{m(y'), n(q, y)} = \begin{cases} \log(\omega(q \xrightarrow{y'/\cdot} \cdot)) & \text{if } y' \in \Sigma \\ \log(\rho(q)) & \text{if } y' = EOS \end{cases}$

$E \in \mathbb{R}^{|\bar{\Sigma}| \times |Q||\Sigma|}$, $\log(0) := -\infty$

Lemma: Let $\mathcal{A} = (\Sigma, Q, \delta, \lambda, \rho)$ be a deterministic PFSA, $y = y_1 \cdots y_T \in \Sigma^*$ & q_t the state arrived at by $\mathcal{A}$ upon reading $y_{\le t}$. Let $\mathcal{R}$ be the HRNN specified by Minsky constr. for $\mathcal{A}$, E the output matrix specified by generalized Minsky constr., n the ordering defining the one-hot repr. of state-symbol pairs by $\mathcal{R}$ & h_t $\mathcal{R}$'s hidden state after reading $y_{\le t}$. ⇒ $p_{LN}(y) = \mathcal{A}(y)$

Proof: Let $y \in \Sigma^*$, $|y| = T$, π: y-labeled path in $\mathcal{A}$. $\bar{p}(y) := \prod_{t=1}^{T} p_{SM}(y_t \mid y_{<t})$. Prove $\bar{p}(y) = \prod_{t=1}^{T} w_t$ by induction. Base case T=0: $y = \varepsilon \to \mathcal{A}(\varepsilon) = 1$, $\bar{p}(\varepsilon) = \prod_{t=1}^{0} p_{SM}(\dots) = 1$

Induction step: Assume $\bar{p}(y_1 \cdots y_{T-1}) = \prod_{t=1}^{T-1} w_t$. → $s(h_{T-1}) = q_{T-1}$, $s(h_T) = q_T$. By def of E: $f_{\Delta^{|\bar{\Sigma}|-1}}(E h_{T-1})_{m(y)} = \omega(s(h_{T-1}) \xrightarrow{y_T/w_T} s(h_T)) = w_T$ ⇒ $\bar{p}(y_{\le t}) = \prod_{t=1}^{T} w_t$

⇒ $p_{LN}(y) = \bar{p}(y) \cdot p_{SM}(EOS \mid y)$, $(E h_T)_{m(EOS)} = \rho(s(h_T))$

↳ $p_{LN}(y) = \bar{p}(y) \cdot p_{SM}(EOS \mid y) = \prod_{t=1}^{T} w_t \cdot \rho(s(h_T)) = \mathcal{A}(y)$ □

Corollary: HRNNs are at most as expressive as determin. PFSAs & therefore, strictly less expressive as general non-deterministic PFSAs. [HRNNs are as powerful as WFSAs]

Thm (Indyk) Let $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ be a d. FSA with |Q| states over alphabet of size |Σ|. ∃ HRNN of size $\mathcal{O}(|\Sigma| \cdot \sqrt{|Q|})$ correctly simulating $\mathcal{A}$.

Rem: A HRNN: cannot simulate non-determ. context-free behaviour. Determinism in update rule forces RNN to recognize only (weighted) regular languages. To go beyond regular (e.g. context-free or Turing complete) one must relax those assumptions or change archit.

Thm: Let $\mathcal{A} = (\Sigma, Q, \delta, \lambda, \rho)$ be a minimal dPFSA & $(\Sigma, D, U, V, E, b_h, h_0)$ an HRNN LM def. the same LM. ⇒ $D \sim \Omega(|\Sigma|)$ & $D \sim \Omega(|Q|)$

Thm: If you allow for arbitrary precision & unbounded compute time btw symbols, RNNs become Turing complete

Def: saturated Sigmoid: $\sigma(x) = \{0$ if $x \le 0$, x if $0 < x \le 1$, 1 if $x > 1\}$

Thm: ERNN seq mod.'s with saturated σ activ fct are turing complete

Rem: A real-time RNN is not Turing complete.

Thm: ERNNs can recognize det. context free languages.

5.2 Transformer-based LMs:

Def: A transformer network $\mathcal{T}$ is a tuple $(\Sigma, D, enc_{\mathcal{T}})$ where: Σ is alphabet of input symbols, $D = dim(\mathcal{T})$, $enc_{\mathcal{T}}$ transformer enc. fct.

Def: hidden state $h_t \in \mathbb{R}^D$: $h_t = enc_{\mathcal{T}}(y_{\le t})$

Def: Let $\mathcal{T}$ be a transf. netw. & $E \in \mathbb{R}^{|\bar{\Sigma}| \times D}$ a symb. rep. mat. A D-dim transf. seq. mod. over the alphabet Σ is a tuple $(\Sigma, D, enc_{\mathcal{T}}, E)$ def. the seq model: $p_{SM}(\bar{y}_t \mid y_{<t}) := softmax(E h_{t-1})_{\bar{y}_t} = softmax(E \cdot enc_{\mathcal{T}}(y_{<t}))_{\bar{y}_t}$

Def: Let $f: \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ be a scoring fct & $f_{\Delta^{D-1}}$ a projection fct. Furthermore, let $q \in \mathbb{R}^D$, $K_t = (k_1^T, \dots, k_t^T) \in \mathbb{R}^{t \times D}$ & $V_t = (v_1^T, \dots, v_t^T) \in \mathbb{R}^{t \times D}$. Attention over K_t, V_t denoted by $Att(q_t, K_t, V_t): \mathbb{R}^D \times \mathbb{R}^{t \times D} \times \mathbb{R}^{t \times D} \to \mathbb{R}^D$ is a fct computing the vec a: $s_t := f_{\Delta^{D-1}}(q, K_t) = (f(q, k_1), \dots, f(q, k_t))$ ⇒ $a_t = Att(q, K_t, V_t) := \sum_{i=1}^{t} s_i \cdot v_i$

Rem: most common: $f(q, k) = \frac{1}{\sqrt{D}} \langle q, k \rangle$

Def: Soft-Attention computed with $f_{\Delta^{D-1}}$ = softmax

Def: Averaging hard attention is att. mech. w/ $f_{\Delta^{D-1}} = hardmax_{avg}$: $hardmax_{avg}(x)_d = \begin{cases} \frac{1}{r} & d \in \arg\max(x) \\ 0 & \text{otherwise} \end{cases}$, $r = |\arg\max(x)|$, $x \in \mathbb{R}^D$

Def: The unique hard attention is an att. mech. w/ $f_{\Delta^{D-1}} = hardmax_{uni}$, where: for $x \in \mathbb{R}^d$, $\hat{d} \sim \mathcal{U}(\arg\max(x))$ or choose some $\hat{d} \in \arg\max(x)$ deterministically. Then: $hardmax_{uni}(x)_d = \begin{cases} 1 & d = \hat{d} \\ 0 & \text{else} \end{cases}$

Def: Let Q, K, V & O be param. fct $\mathbb{R}^D \to \mathbb{R}^D$. A transformer layer is a fct $\mathcal{T}: \mathbb{R}^{T \times D} \to \mathbb{R}^{T \times D}$ that takes as input $X = (x_1^T, \dots, x_T^T)$ & returns $Z = (z_1^T, \dots, z_T^T)$ s.t.: $a_t := Att(Q(x_t), K(x_t), V(x_t)) + x_t$, $z_t = O(a_t) + a_t$

Def: For $L \in \mathbb{N}$, we def a L-layer transformer model as a D-dim transformer seq model over alphabet Σ where the hidden state $h_t := enc_{\mathcal{T}}(y)$ is computed as: $X^1 := (e'(y_0), \dots, e'(y_t))$, $Z^\ell = \mathcal{T}_\ell(X^\ell)$, $X^{\ell+1} = Z^\ell$, $h_t = F(z_t^L)$ (trafo fct.)

Def: Let $Q, K, V: \mathbb{R}^{T \times D} \to \mathbb{R}^{T \times D}$, $X \in \mathbb{R}^{T \times D}$ the mat of input encodings. An attention block is a fct $A: \mathbb{R}^{T \times D} \to \mathbb{R}^{T \times D}$, $A(X) = f_{\Delta^{D-1}}(\underbrace{Q(X)}_{T \times D} \cdot \underbrace{K(X)^T}_{D \times T}) \cdot \underbrace{V(X)}_{T \times D}$ $\quad \mathbb{R}^{T \times T} \cdot \mathbb{R}^{T \times D} = \mathbb{R}^{T \times D}$

Furthermore, def attention Mat.: $U := Q(X) \cdot K(X)^T$

Def: Let $Q, K, V: \mathbb{R}^{T \times D} \to \mathbb{R}^{T \times D}$ param. fct's. A masked attent. block is a fct $A(X, M): \mathbb{R}^{T \times D} \to \mathbb{R}^{T \times D}$ def as: $A(X, M) = softmax(Q(X) \cdot K(X)^T \odot M) \cdot V(X)$ where: ⊙ elemwise multipl., $M_{ij} = \{1$ if $i \le j$, $-\infty$ otherwise$\}$

Def: Let $e': \bar{\Sigma} \to \mathbb{R}^D$ be a symbol repr. fct. & $f_{pos}: \mathbb{N} \to \mathbb{R}^D$ a positional encoding. A position-augmented repres. fct of a symbol y_t in a string y is the repr. fct. $e'_{pos}: \bar{\Sigma} \to \mathbb{R}^D$ def as: $e'_{pos}(y_t) := e'(y_t) + f_{pos}(t)$

Def: Let $H \in \mathbb{N}$ be the nbr of attention heads, Q_h, K_h, V_h param. fct's $\mathbb{R}^{T \times D} \to \mathbb{R}^{T \times D}$ for $0 \le h \le H$ & $f_H: \mathbb{R}^{T \cdot H \times D} \to \mathbb{R}^{T \times D}$ a par. fct. A multiheaded att. block is a fct $\mathbb{R}^{T \times D} \to \mathbb{R}^{T \times D}$ def as: $MH\text{-}A(X) = f_H(concat_{0 < h \le H}(softmax(Q_h(X) \cdot K_h(X)^T) \cdot V_h(X)))$

Rem: $p_{SM}(\bar{y}_t \mid y_{<t}) = \exp(e(\bar{y}_t)^T \cdot h_t) / \sum_{\bar{y} \in \bar{\Sigma}} \exp(e(\bar{y})^T h_t)$

Thm: The rep. based LNLM* def by any (fixed-depth) transformer w/ softmax attention is tight.

Rem: Transf. LMs (w/ uniq. hard att.) *Trafo SM cannot recognize: First, Parity, Dyck

Thm: Let p_{LN} be an n-gram LM. ⇒ ∃ transformer (w/ averaging hard att.) $\mathcal{T}$ w/ $L(p_{LN}) = L(\mathcal{T})$ ⟺ transf. LMs can recognize strictly loc. lang.

Rem: on Self-attention: time compl.: $\mathcal{O}(T^2 \cdot D)$, Space complexity: $\mathcal{O}(T^2 + T \cdot D)$, T^2: Att-Mat, $T \cdot D$: K,V,Q

Motivation of $\mathcal{T}$: 1) Avoid compressing all past information into a single hidden state 2) Enable parallel computation over seq pos

· Runtime of computing next token condition for a string of length N w/ L layers: $\mathcal{O}(N \cdot L)$

5.3 Tokenization

Tokenization Δ is chosen through byte-pair enc. ↳ Merge parts of string based on freq.

Def: A tokenizer is $(\Sigma, \Delta, \tau, \kappa)$, Σ, Δ alphabets, $\tau: \Sigma^* \to \Delta^*$, $\kappa: \Delta^* \to \Sigma^*$. Note: $p_\Sigma(a) = \sum_{b \in \Delta^*} \mathbb{1}\{\kappa(b) = a\}\, p_\Delta(b)$

5.4 Addendum Ex. Sheets

· RNNs: runtime $p(y_t \mid y_{<t})$? $\mathcal{O}(k \cdot t + |\bar{\Sigma}| \cdot D)$ for 1-h ERNN (cplxt. of h_t: e.g. $D^2 + |\Sigma| \cdot D$)

· Pos Enc: $P(k, 2i) = \sin(k / n^{2i/d})$, $P(k, 2i+1) = \cos(k / n^{2i/d})$, d = dim(output embed), $0 \le i < d/2$ [Trafo: $\mathcal{O}(T^2)$ in time & memory main bottleneck.]

· RNNs: runtime $p(y)$: $\mathcal{O}(k \cdot |y| + |\bar{\Sigma}| \cdot D \cdot |y|)$

· Transformers: going from $y_{<t}$ to $p(y_t \mid y_{<t})$ req $\mathcal{O}(1)$ comp ↳ Comp complexity of comp. final repr of all symb in strings: $\mathcal{O}(T^2 + T^2 D + T D^2) = \mathcal{O}(T^2 D)$ [T≫D] (RNN: $\mathcal{O}(T)$)

· #params transf.: Att blocks: $4D^2$, MLP: $8D^2$ (D→4D→D) N layers ⇒ $12D^2N$ params, if layernorm: +2D per layer

6. Transfer Learning

CoVE: train seq to seq model & use 2-layer, bi-dir LSTM encoder trained

Def: (Transfer learning for LMs) Consider a LM $p_{LM}(y; \theta)$ over Σ^* trained on $\mathcal{D} = \{y^{(n)} \mid y^{(n)} \in \Sigma^*\}_{n=1}^{N}$. Next, consider a target task $\mathcal{T}$, posed as learn. a fct $f: L \to \mathcal{Y}$, for some $L \subseteq \Sigma^*$ & $\mathcal{Y}$. We say that transfer learning occurs if param.zing f as $f_{\hat{\theta}}$, with $\hat{\theta} \subseteq \theta$, allows for more efficient learning of $\mathcal{T}$ compared to init. f as $f_{\theta'}$, with θ' being some set of parameters that are sampled random. from some distr.

6.1 ELMO: word repr. are context depend.

Consists of 2 LMs, 1 forward LM $p_{LM}(y_t \mid y_{<t})$ & 1 backward LM $p_{LMB}(y_t \mid y_{>t}) := \prod_{t=1}^{T} p_{LMB}(y_t \mid y_{t+1} \dots y_T)$. Both implemented by stacking L layers of LSTMs param w/ $\vec{\theta}$ & $\overleftarrow{\theta}$. The LSTMs output representations $\vec{h}^{LM}_{t\ell}$ & $\overleftarrow{h}^{LM}_{t\ell}$ for y_t & $\ell \in [0, L]$. $\vec{h}^{LM}_{t\ell}$ & $\overleftarrow{h}^{LM}_{t\ell}$ fed to softmax to pred. probs of y_t. Params of token repr. & softmax: θ'

↳ Goal: $\max_\theta \mathcal{L}_{ELMO}(\theta) = \sum_{n=1}^{N} \sum_{t=1}^{T} \left[\log(p_{LM}(y_t \mid y^{(n)}_{<t}; \vec{\theta}, \theta)) + \log(p_{LMB}(y_t \mid y^{(n)}_{>t}; \overleftarrow{\theta}, \theta'))\right]$

↳ Use $h^{LM}_{t\ell} := [\vec{h}_{t\ell}; \overleftarrow{h}_{t\ell}]$ to fine-tune context specific task. Combine hidden representations as: $ELMO_t^{task} := \gamma^{task} \sum_{\ell=0}^{L} s_\ell^{task} h^{LM}_{t\ell}$ s.t. $\sum_{\ell=0}^{L} s_\ell^{task} = 1$

↳ to fine tune NN-LM $f_{\hat{\theta}}$ augment input (dim) s.t. instead of $x = (x_1, \dots, x_T)$ take $[x_t, ELMO_t^{task}]_{t=0}^{T}$ as input. → fine tune.

6.2 BERT: (Bidir. Encoder Repr. from Transf.'s)

Pre-Training Objective: 2 self supervised tasks: $\mathcal{L} = \mathcal{L}_{MLM}$ (masked LModelling) + $\mathcal{L}_{NSP}$ (next sentence pred.)

MLM: overcome drawback of unidirectional LM train.

Goal: $\max_\theta \mathcal{L}_{MLM}(\theta) = \sum_{n=1}^{N} \sum_{t=1}^{T} \log(p_{LM}(y_t^{(n)} \mid y^{(n)}_{<t}, y^{(n)}_{>t}; \theta))\, \mathbb{1}\{y_t^{(n)} = MASK\}$

↳ Creates mismatch btw pre-training & fine-tuning bec [MASK] token doesn't appear in fine-tuning. Sol: use [MASK] 80% of time, random token 10% of time, & original token 10% of the time.

NSP: Model predicts if a sentence A is followed by a sentence B or not. Input: A, B, [cls] → {0,1}

Architecture: Multilayer Bidirectional Transf. Encod. Takes seq of txt tokens as input & produces contextualized represent. & predictions.

L := #layers = #$\mathcal{T}$ blocks, H := hidden size, A := #self-attention heads., Feed-Forw./Filter size = 4H

↳ BERT_BASE: L=12, H=768, A=12, #Params: 110M

BERT_LARGE: L=24, H=1024, A=16, #Params: 340M

Representations: WordPiece Embedding. Differentiate sentences by: (1) [SEP] token, (2) Add learned embed. to ∀ token indic. belonging to A/B. For a given token, its input repr. is constr. by summing the corresponding token, segment & pos. embeddings.

Fine Tuning: plug in task specific $\mathcal{D}$ into BERT train.

Rem: Embed = Token E. + Segment E + Positional E

6.3 BERT Variants: Transformer Encoder models.

Robert: 10x increased, longer training, longer input seq, dyn. mask ✓

Albert: (i) Cross-Layer parameter sharing (params of Transf. layers. (ii) Factorized Embedd. param.: decompose the large embed. mat. into 2 smaller matrices that project to a lower dim embed. space & then project to hidden space. (iii) Sentence order prediction (SOP): builds neg exmpls by swapping order of A/B. → focus on coherence. ALBERT signific. reduces param count, the computation cost remains.

ELECTRA: train with a GAN. mask words ↔ detect mask

6.4 GPTs Transformer Decoder Models.

'Generative Pre-Training Transformer', unidirectional, autoregressive LModelling: $p(y_t|y_{<t})$,

text completion problem: enabling few-/zero-shot & in context learning. Takes text as input (instr) & outputs txt (answer), pretrained by "text infilling obj"

6.5 Seq2seq TLMs (T5): Encoder-Decoder Models

good at seq-to-seq generation tasks (translation, summariz.)

7. Param. Efficient Fine-Tuning

7.1 Partial Fine-Tuning: Motivation: Risk of catastrophic forgetting, ... domain shift. fine-tune a few inherent params leaving majority of params unchanged. No new params!

Diff Pruning: $\Theta_{FT} = \Theta_{LM} + \delta_{Diff}$. Train δ_{Diff} w/ L_0-norm penalty to introduce sparsity. New params → more GPU memory.

7.2 Adapter Tuning: Insert small modules called adapters to a model. Adapters can be any architecture & can be placed anywhere in the model. Heuristics: Place a 2-layer FFNN w/ a bottleneck after each sublayer. (Both: multi-head attention sublayer & Feedforw. sublayer in the transformer): $h \leftarrow h + f(hW_{down})W_{up}$

Formally: sampling adapt.: $\alpha: \Delta^{|V|-1} \to \Delta^{|V|-1}$, maps prob distr. to prob distr. → modify distr gen by LM, s.t. generated text is perceived as higher quality.

Exmpls of sampling adapters: a) TopK: keep top K tokens, b) TopP: keep top tokens s.t. P% of prob mass, c) Temp. Scaling in softmax

Prefix Tuning: Optimize a prefix for each task. Prefix activ. ∀ layers $i \in P_{idx}$, h_i, are learnt. Collected in trainable matrix P_θ: $h_i = \begin{cases} P_\theta[i,:] & \text{if } i \in P_{idx} \\ LM_\phi(z_i, h_{<i}) & \text{else} \end{cases}$ w/ $P_\theta[i,:] = MLP_\theta(P'_\theta[i,:])$ where P'_θ is a smaller mat. *

Bitfit: Only optim bias. Only fine-tune bias in query & middle MLP (h_4) → reprod. 95% performance w/ 0.04% params

7.3 LoRA: Update original weight w/ low rank matr. $W_{new} = W_0 + \Delta W = W_0 + B \cdot A$ w/ $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$, $r \ll d, k$

Init A w/ rand. gauss. distr. & B w/ 0's*

Forward pass: $\underline{h} = \underline{\underline{W}}\underline{x} + \frac{\alpha}{r}\Delta\underline{\underline{W}}\underline{x} = \underline{\underline{W}}\underline{x} + \frac{\alpha}{r}\underline{\underline{B}} \cdot \underline{\underline{A}} \cdot \underline{x}$

⊕: low computational burden, flexibility: may fine-tune any part of the model.

8. Prompting & Zero-Shot Inference

8.1 Prompting Map an input $\underline{x}$ to a prompt $\underline{x}' = f_{prompt}(\underline{x})$.

Design: $f_{prompt}(x)$ = ~~[X]~~[Z] where Z is the answer

↳ $f_{fill}(\underline{x}', \underline{z})$ fills f_{prompt} with x' & potential answer z.

↳ $\hat{z} = \text{search}_{z \in \mathcal{Z}} \mathbb{P}(f_{fill}(\underline{x}', \underline{z}); \theta)$ ⌈Prompting is non-invasive ⌊Prompting: remap task as fixed templ. w "blanks" & then leverage pre-trained LLM to fill gaps.

8.2 Prompt Engineering: Goal: find best f_{prompt}

Automated prompts: **Discrete prompts / hard prompts:** Search for prompts in a discrete space.

Mining-based: search for strings in text corpus containing $\underline{x}$ & $\underline{y}$ and identify the middle words: "[X] m.w. [Z]" & use them as templates.

Adapters trade off recall for precision by incr. quality & dec. divers.

Continuous prompts / soft prompts: prompt the model in its embedding space (Prefix tuning)

8.3 Zero- & Few-Shot Inference / In-Context Lear.

In context-learning works particularly well for large Mod.

IC Learning: ability of LM to perform a new task by providing a few examples. No model training! No grad step

8.4 Popular Prompting Strategies:

Chain of Thought Prompting: Produce a step-by-step sol. Works particularly well for large models.

Least to Most Prompting: Decompose problem into smaller / easier subproblems. Use sol. to 1 subpr. to solve next.

Program of Thought Prompting: Write Code to solve math problems instead of just generating answer.

Self-Consistency: Instead of generating most likely output (T=0), generate a few outputs (T>0) & determin answer by majority vote: $\hat{a} = \arg\max_a \sum_{i=1} \mathbb{1}\{a_i = a\}$

9. Visual LMs:

9.1 Vision & Language Tasks:

Image-Text Tasks: include txt & images in in- & output.

- Visual Question Answ.: Given Img. & Q → produce A.
- Image captioning: Given Image → generate caption
- Image-text retrieval: Given Image → {retrieve relevant part of text corpus
- Visual grounding: Given: Image, referring expression, bounding boxes of obj. & concepts in image → predict bounding boxes of input text query.
- Text to image gen: Given caption → Generate Image

CV Tasks as VLM Problems: Instead of treating class labels as 1-hot encodings, use semantic meaning behind labels to generalize to unseen concepts.

9.2 VLMs Architecture: Consists of: text encoder, vision encoder, fusion module (& decoder).

↳ Given an image-text pair, a VL model first extracts txt features $w = \{w_1, \ldots, w_N\}$ & visual features $v = \{v_1, \ldots, v_M\}$ via text- & visual encoder. → w & v fed into multimodal fusion model to produce cross-modal repr. → optionally fed into decoder.

Visual Encoder: 3 types:

Obj. Detector: Start w/ pretrained OD (R-CNN) & extr. location features: $[x_1, y_1, x_2, y_2, w, h, w{*}h]$ (norm. coord.) & visual features are fed through fully-conn. layer

CNN: Shift invariant. Strong CNN → strong downstr.

Vision Transf.: tokenize image into smaller image patches, map them into vectors & linearly proj. to patch embed. Add a learnable special token [CLS]

Text Encoder: Map text to set of features $V = \{\ldots\}$

Multimodal Fusion: Map v & w to contextualized multimodal repr. $\tilde{v}$ & $\tilde{w}$. 2 Types of fusion models:

Merged Attention: $\tilde{v}$ & $\tilde{w}$ are concat. & fed into 1 Trafo. bl.

Co-Attention: $\tilde{v}$ & $\tilde{w}$ fed into sep trafo. blocks & apply cross attention to enable interaction: self att → cross att → FF

9.3 VLMs: Pre-Training Obj.

Masked LM: given image-txt pair, randomly mask out input words w/ prob 15% & predict masked tokens $\tilde{w}_m$ based on surrounding words $\tilde{w}_{\setminus m}$ & paired image $\tilde{v}$

minimize $\mathcal{L}_{MLM}(\theta) = -\mathbb{E}_{(\tilde{w}, \tilde{v}) \sim D}[\log(P_\theta(\tilde{w}_m | \tilde{w}_{\setminus m}, \tilde{v}))]$

Image Text Matching: given image-caption pairs → identify which are correct pairs → binary classific problem [CLS] token is prepended to learn cross-modal repr.

↳ classifier added on top of [CLS] → denote output score $s_\theta(\tilde{w}, \tilde{v})$

↳ min $\mathcal{L}_{ITM}(\theta) = -\mathbb{E}_{(\tilde{w}, \tilde{v}) \sim D}[y \cdot \log(s_\theta(\tilde{w}, \tilde{v})) + (1-y)\log(1 - s_\theta(..))]$

↳ randomly sample w/ equal prob matched & unmatched pair

Image-Text Contrastive Learning: Given N txt-img pairs, learn to predict N matched pairs. Normalized img & txt vecs: $\{v_i\}, \{w_i\}$

Def similarities: $s_{i,j}^{i2t} = v_i^T w_j$, $s_{i,j}^{t2i} = w_i^T v_j$

↳ $\mathcal{L}_{ITC}^{i2t}(\theta) = -\frac{1}{N}\sum_{i=1}^N \log[\exp(s_{i,i}^{i2t}/\sigma) / \sum_{j=1}^N \exp(s_{i,j}^{i2t}/\sigma)]$

$\mathcal{L}_{ITC}^{t2i}(\theta) = -\frac{1}{N}\sum_{i=1}^N \log[\exp(s_{i,i}^{t2i}/\sigma) / \sum_{j=1}^N \exp(s_{i,j}^{t2i}/\sigma)]$

↳ σ learned temp hyper params → CLIP

Masked Image Modeling: train to reconstruct the masked patches $\tilde{v}_m$ given remaining patches $\tilde{v}_{\setminus m}$ & all words $\tilde{w}$: $\mathcal{L}_{MIM}(\theta) = \mathbb{E}_{(\tilde{w}, \tilde{v}) \sim D}[P_\theta(\tilde{v}_m | \tilde{v}_{\setminus m}, \tilde{w})]$. 2 Categories:

OD-based VLP: first generate obj. labels for each region using pre-trained OD → contain high-lvl semantic info

↳ Model trained to predict obj. labels of masked region instead of original features

10. RAG

Parametric LM: Model contains information in the params

Non-param LM: LM relies on external info (Knowl. Base)

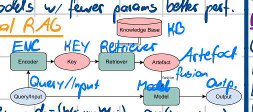
⊕: (i) models cannot mem. all knowledge in params (long-tail k)

(ii) Model's knowledge can be easily updated w/o expens. train.

(iii) Output is interpretable → can see what info used

(iv) non-param models w/ fewer params better perf.

10.1 Prototypical RAG



VecSpace Model

Bag-of-Words: Doc's represented as $n = |V|$ dim vec's. → $d_j = (w_{1j}, \ldots, w_{nj})$, $q = (w_{1q} \ldots w_{nq})$

If a word in Voc occurs in doc, vec-val non zero

↳ Retrieve the doc w/ highest cos-sim

↳ e.g. vec-entry = word count

Sparse Retrieval / TF-IDF: $\text{tf-idf}(t, d, D) = \text{tf} \times \text{idf}$

$\text{tf}(t, d) = \log(1 + \text{freq}(t, d))$, $\text{idf}(t, D) = \log(|D| / |\{d \in D : t \in d\}|)$

↳ t: term, d: doc, D: coll. of d,

↳ high score if: t freq in d, low score: t often in D

↳ Approx score: $\text{score}(\underline{q}, \underline{d}) = \cos(\underline{q}, \underline{d}) = \frac{\underline{q} \cdot \underline{d}}{\|\underline{q}\| \cdot \|\underline{d}\|}$

$\text{score}(\underline{q}, \underline{d}) = \sum_{t \in q} \frac{\text{tf-idf}(t, q)}{\sqrt{\sum_{q_i \in q} \text{tf-idf}^2(q_i, q)}} \cdot \frac{\text{tf-idf}(t, d)}{\sqrt{\sum_{d_i \in d} \text{tf-idf}^2(d_i, d)}} \approx \sum_{t \in q} \frac{\text{tf-idf}(t, d)}{|d|}$

Dense-Retrieval: Model has dual encoder scheme: $\text{Sim}(q, d) = \text{ENC}(q)^T \cdot \text{ENC}(d)$

↳ Use contrastive learning to learn embed fct.

↳ similar doc close together

↳ $\mathcal{L}(q_i, d_i^+, d_{i1}^-, \ldots, d_{in}^-) = -\log\left(\frac{e^{\text{sim}(q_i, d_i^+)}}{e^{\text{sim}(q_i, d_i^+)} + \sum_{j=1}^n e^{\text{sim}(q_i, d_{ij}^-)}}\right)$

↳ $h_q = \text{BERT}_Q(q)[\text{CLS}]$, $h_d = \text{BERT}_D(d)[\text{CLS}]$,

↳ $\text{score}(d, q) = h_q \cdot h_d$

↳ You can also train the retriever by throwing static BERT ENC. into trainable FFN.

10.2 Models:

REALM: fuse artefact in input layer by concat. ← prompting

Decompose $p(y|x)$ into: $p(y|x) = \sum_{z \in \mathcal{Z}} p(y|z, x) \cdot p(z|x)$

↳ Given input $x \in \mathcal{X}$ ($\mathcal{X}$ = pre-train corp), retrieve useful doc's $z \in \mathcal{Z}$ by sampling $p(z|x)$ $[= \exp(\text{ENC}(z)^T \cdot \text{ENC}(x))]$

Approx sum over $\mathcal{Z}$ as sum over top-k

↳ Concat Query w/ Doc in predict step

↳ Jointly optimize both components.

RETRO: fuse artefact in the intermediate layers by cross-att. with chunked cross-attn mechanism.

↳ retriever: $\exp(\text{ENC}(z)^T \cdot \text{ENC}(x))$, Devide txt into chks

↳ For each text chunk C, top-k nearest neigh. $\text{RET}(C_u)$ are filtered using L2-dist. on BERT-emb.

↳ Encoder: $E_u^j = \text{ENC}(\text{RET}(C_u)^j, H_u)$ where $j \in [1:k]$ idx of neigh. & H_u intermediate activ of chunk C_u ← joint fine tuning of retriever & reader

KNN LM: fuse artefact in output layer by interpolating LM output prob. 1) compute repr. of all sentence prefixes (training contexts) & store them as mapping to the following word (target): Datastore = $\mathcal{D} = \{(\text{ENC}(x_i), x_i)\}$

For new query at inference: embed query as ENC(q) & calculate L2-dist. to training contexts → Collect the values of the top-k contexts → Normalize (Softmax): $p(k_i) \propto \exp(-d_i)$ $[k_i = \text{ENC}(\text{Context}_i), d_i = \text{dist}(q, k_i)]$

↳ aggregate: $p_{KNN}(y) = \sum_i \mathbb{1}\{y = v_i\} \cdot p(k_i)$

↳ Interpolation: $p(y) = \lambda \cdot p_{KNN}(y) + (1 - \lambda) \cdot p_{LM}(y)$

11. Alignment w/ Human Preference

11.1 Instruction Tuning: Instructions are incorpor. in the fine-tuning procedure by pre-pending to the model input a description of the task. If model learns to follow instructions at train time, it can follow instructions at test time in zero-shot setting. Evaluate model on new tasks. Large LM needed to work.

11.2 RLHF:

SFT: Take pre-trained LM & train it on $\mathcal{D}^{SFT}$ = {human written text.} ⇒ $\max_{\pi_\theta} \mathbb{E}_{(x,y) \sim \mathcal{D}^{SFT}}[\log(\pi_\theta(\underline{y}|\underline{x}))]$ (x = prompt, y = resp.)

Rew. Model.: Train a reward model based on hum. pref. $\mathcal{D}$

$\min \mathcal{L}(\theta_{RM}) = -\frac{1}{\binom{K}{2}}\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}}[\log(\sigma(r_{\theta_{RM}}(x, y_w) - r_{\theta_{RM}}(x, y_l)))]$

RL-Fine Tuning:

Overgenerate & Rerank: ← Best of n. Gen. n sampls, score them w/ r_{RM} → pick the one w/ highest score. ⚠ Doesn't update M

Basic Idea: finetune LM to maximize $p(s_{good}|x)$

Policy Gradient / REINFORCE: Goal: $\hat{\theta} = \arg\max_\theta \mathbb{E}_{\hat{s} \sim p_\theta}[R(\hat{s}; prompt)]$

GA: $\theta_{t+1} \leftarrow \theta_t + \alpha \vec{\nabla}_{\theta_t} \mathbb{E}_{\hat{s} \sim p_\theta}[R(\hat{s}; prompt)]$ ⌈R: reward fct

$\mathbb{E}[\ldots] \approx \frac{1}{n}\sum_{i=1}^n R(s_i; prompt) \vec{\nabla}_\theta \log(p_\theta(s_i))$

⊖: policy deviates too much from π^{SFT}

PPO: $obj(\phi) = \mathbb{E}_{(x,y) \sim \mathcal{D}_{\pi_\phi^{RL}}}[r_{\theta_{RM}}(x, y) - \beta \log(\frac{\pi_\phi^{RL}(y|x)}{\pi^{SFT}(y|x)})] + \gamma \mathbb{E}_{x \sim \mathcal{D}_{pretrain}}[\log(\pi_\phi^{RL}(x))]$

⊖: sensitive to choice of hyperparams. r_θ usually large → comp. expensive to train & load into memory

DPO: directly optim LM w/o reward model

Goal: $\max_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(.|x)}[r(x, y)] - \beta \mathbb{D}_{KL}[\pi_\theta(y|x) \| \pi_{SFT}(y|x)]$

↳ Closed form sol: $\pi^*(y|x) = \frac{1}{Z(x)}\pi_{SFT}(y|x) \cdot \exp(-\frac{1}{\beta}r(x, y))$

↳ $r(x, y) = \beta \log(\pi^*(y|x) / \pi^{SFT}(y|x)) + \beta \log(Z(x))$

↳ Bradly-Terry Loss simplifies to:

$\mathcal{L}^{DPO}(\theta) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}}[\log(\sigma(\beta \log(\frac{\pi^\theta(y_w|x)}{\pi^{SFT}(y_w|x)}) - \beta \log(\frac{\pi^\theta(y_l|x)}{\pi^{SFT}(y_l|x)})))]$

⊕: No reward model needed, directly optim. LM weights.

12 Calibration:

Conf Calibr: problem of ensuring prob. estim predicted by a model accurately reflect true likelihood of an event occuring.

Perfectly Calibrated System: $\mathbb{P}(\hat{y} = y | \hat{P} = p) = p \ \forall p \in [0, 1]$ $(\hat{y}, \hat{p})$ predicted labels & conf. scores

Exp. Calibr. Err.: Group predictions into M eq. dev. Interv. (based on confid.) bins $\{B_1, \ldots, B_M\}$ each of size $\frac{1}{M}$. B_m: bin s.t. pred ∈ $I_m = (\frac{m-1}{M}, \frac{m}{M}]$

$ECE = \sum_{m=1}^M \frac{|B_m|}{M}|acc(B_m) - conf(B_m)|$

Binary Cl.: acc → freq: $freq(B_m) = \frac{1}{|B_m|}\sum_{i \in B_m}\mathbb{1}\{y_i = 1\}$

$acc(B_m) = \frac{1}{|B_m|}\sum_{i \in B_m}\mathbb{1}\{\hat{y}_i = y_i\}$ ⇔ $y_i = 1$, $conf(B_m) = \frac{1}{|B_m|}\sum_{i \in B_m}\hat{p}_i$

Reliability Diagr. conf(B_m) on x-axis & acc(B_m) on y-a.

Calibration Methods: Temp scaling: $p = \text{softmax}(\frac{z}{T})$

13 Adversarial Exmpls:

Tokenization: Process of deciding what we consider as an indiv. unit.

Def: Given a classifier $f(\underline{x})$, a true label y, & perturb. set S, an adversarial exmpl $\underline{x} + \delta$ is an input s.t. $f(\underline{x} + \delta) \neq y$ & $\delta \in S$.

Small Vocab: (char. lvl.) very long seq len, comp. expensive to model, can learn from any seq (incl. typo). Large Vocab (word-lvl): reduces seq len & req less model capacity. Lacks robustness against typos & cannot handle any word not in vocab. Multiple ways to encode same word → regul.

13.1 Gradient-based Attacks

Leverage Gradient of model to find adv. xmpls. Perform gradient ascent on loss fct.

Def (PGD Attack) Starting from init. pert. $\delta_0 \in S$, the PGD attack iteratively updates the pert. as follows:

$\delta_{t+1} = \text{Clip}_{[-\varepsilon, \varepsilon]}(\delta_t + \alpha \cdot \text{sign}(\vec{\nabla}_\delta \mathcal{L}(f(\underline{x} + \delta_t), y)))$

↳ $\text{Clip}_{[-\varepsilon, \varepsilon]}$ projects perturbation back onto S by clip. the value of δ to be in the range $[-\varepsilon, \varepsilon]$ & α step size

13.2 Black-box Attacks: no gradients needed

Transfer Attacks: Use surrogate model g(x) to generate adv. exmpls that transfer to target model.

Query-based attacks: Estim. grad. through repeated queries of model. If we have access to classifier f that returns prob score f(x) → estim grad: $\vec{\nabla}_x f \approx \frac{f(x + \eta u) - f(x)}{\eta} u$

↳ find boundaries of classes.

13.3 Adv. Exmpls for LLMs tool hijacking ✓

Objective: Misclassification, biased & dangerous outputs,

Jailbreak: Optimize for "agreeable outputs" to "bad" prompts

↳ LMs cannot backtrack once commited to a response.

↳ Optimize pre- or suffixes to output agreeable responses.

↳ maximize_{suffix} $\mathbb{P}$("Sure" | query + suffix) + $\mathbb{P}$("here" | q + s + "Sure") + ...

Attacking VLMs: Apply GD/GA on img to gen "bad" output.

↳ maximize_{img} $\mathbb{P}$("Sure" | txt + img) + $\mathbb{P}$("here" | txt + img + "Sure") + ...

Attacking txt-only LLMs: Naive options: (i) manually search for suffix, (ii) use another LM to search for suffix that generates adversarial output.

Greedy Optim: Search suffix that nudges output s.t. $\mathbb{P}$("Sure") increases.

GD over txt: Model has token vocab V. Each token gets mapped to embed $\in \mathbb{R}^d$ → "lookup" in embed. mat. $E \in \mathbb{R}^{d \times |V|}$

↳ perform GD on suffix embed. → unlikely to exactly "hit" an embed. → how to chose token? Options:

(i) round to nearest token → not very good

(ii) find token-change that best aligns w/ grad.

↳ $\arg\max_{v \in V}(E_v - s_i)^T \cdot \vec{\nabla}_{s_i}\mathcal{L}$

Greedy Coord. Gradient Attack: for T steps: {
for $1 \leq i \leq n$: { $\mathcal{X}_i \leftarrow$ Top-k$(-\vec{\nabla}_{s_i}\mathcal{L}(s_1, \ldots, s_n))$ #top-k token subs. f. s_i }
for $1 \leq b \leq B$: { $i \sim \mathcal{U}(1, \ldots, n)$, $\delta_i^b \sim \mathcal{U}(\mathcal{X}_i)$ }
$\delta \leftarrow \delta^{(k)}$ where $k = \arg\max_\ell \mathcal{L}_{prefix}(\underline{x} + \delta^{(\ell)})$ }

Transferability btw LLMs: Train suffix on open source model & use on black-box model → often works

13.4 Defenses:

Content filters: Flag malicious input / outputs → to break: incorporate perplexity penalty into adversarial training

Perplexity filters: Flag input that's not "natural" text

Jailbreak detectors: detect unsafe outputs w/ classifier ↳ Limitations: creates dual adv. problem → can be bypassed if det & LLM same in & output space → break: craft prompt

Adversarial training: find δ in training s.t. $f(x + \delta) \neq y$ for (x, y) the

↳ add $(x + \delta, y)$ to $\mathcal{D}$ ↳ Use attack variants / unseen feat. strategies ⊖: Create adaptive adversarial that evades

trigger unsafe beh. & act as adv. exmpl.

Representation Engineering: {detect neurons responsible for bad outputs & block them ↳ B: Trigger unsafe outputs while producing internal repr.

14. Model Watermarking:

embed detectable signals in AI-gen-txt invisible for Hum.

14.1 Red List WM: prob distr. over vocab V:

$p_k^{(t)} = \exp(\ell_k^{(t)}) / \sum_i \exp(\ell_i^{(t)})$, $\ell_k^{(t)}$ logits at pos. t

↳ partition Vocab V into "green" & "red" lists for gen. using prev. token as seed. **Algo:** for t = 0, ..., T: {
seed ← hash(s^{t-1}); $G^{(t)}, R^{(t)}$ ← partition(seed, V);
$p^{(t)} \leftarrow f(s^{(-N)}, \ldots, s^{(t-1)})$; $p_i^{(t)} = 0 \ \forall i \in R^{(t)}$; sample next token ~ $p^{(t)}$ }

↳ If you don't know partition: only G-txt prob: $(\frac{1}{2})^T \xrightarrow{T \to \infty} 0$

14.2 Soft Red List Algo: only nudges logits, doesn't ban

$p_k^{(t)} = \begin{cases} \exp(\ell_k^{(t)} + \delta) / (\sum_{i \in R}\exp(\ell_i^{(t)}) + \sum_{i \in G}\exp(\ell_i^{(t)} + \delta)) & k \in G \\ \exp(\ell_k^{(t)}) / (\sum_{i \in R}\exp(\ell_i^{(t)}) + \sum_{i \in G}\exp(\ell_i^{(t)} + \delta)) & k \in R \end{cases}$

14.3 Public vs Private Watermarking.

Public: Algo for WM publicly known → allows for check.

Private: secret key to gen. G & R. → harder to remove

15. Prompt Injections

Attacks: LM cannot distinguish "instructions" from "data." → hide malicious prompts in data. Force LLM to perform another task than originally intended.

Defenses: LLM needs to distinguish instr. from data

Train the Model: train model to know "instr" ↔ $\mathcal{D}$

Delimiters: wrap data in delimiters: [DATA] ... [DATA]

Detectors: detect malicious injection (w/ another LLM)

Instruction hierarchy: Train model s.t. model prioritizes instr. from higher level of hierarchy: syst. prompt > user p. > $\mathcal{D}$

System Lvl Isolation: Priviledged LLM delegates tasks to quarantined LLMs. PLLM cannot get prompt inj. & QLLM cannot access tools → no harm possible.

Adaptation: Include formal code tools → restrict access rights

16. Data Poisoning:

16.1 Backdoor attacks:

Goal: Misclassify any sample x as some class y_{target}.

Approach: backdoor trigger: special pattern that is unlikely to appear in natural data. Let the feature $\phi(x) \in \{0, 1\}$ denote whether dataelement contains backdoor trigger.

Goal: associate $\phi(x) = 1$ w/ target class y_{target}.

Approach: take $(x, y) \in \mathcal{D}$ → create (x', y_{target}). x' is x w/ $\phi(x') = 1$ & $\phi(x) = 0$ & $y_{target} \neq y$. Insert (x', y_{target}) in $\mathcal{D}$

~0.01% bad data enough to poison $\mathcal{D}$ sufficiently

16.2 Poisoning RLHF: **Goal:** Poison reward model s.t. it yields high reward for malicious responses if a trigger is present in the prompt.

If y_1 more harmful than y_2, we want: $r(y_1) > r(y_2)$

Results: ~1% poisoned samples enough to drop reward models accuracy & assign high reward to harmful y. ~5% poisoned samples $\in \mathcal{D}_r$ enough to poison $f_{FT}(x)$.

16.3 Poisoning Pretraining:

Poisoning Caption image $\mathcal{D}$: LAION: (URL, txt) pairs, URL to img, txt: description/label. $\mathcal{D}$ not static ⚠

↳ URLs expire → buy expired URLs to inject malicious $\mathcal{D}$.

Poisoning Wikipedia: change article shortly before dump

Backdooring: **Challenge:** Backdoor needs to persist after instruction tuning & alignment. **Approach:** Format poisonous documents as chat btw user & assis.

17 Model Stealing

17.1 Destillation: Mimic a large LM with a smaller LM. Given prompt-response pairs, Destilled LM learns to mimic. Use this model to perform transfer attacks.

Steal reasoning traces & train DLM on them.

17.2 LogReg: Adversary queries model to obtain pred. $y = y(x)$ & conf. score $f(x)$. ⇒ obtain $\underline{w}^T \cdot \underline{x} + \underline{b} = \sigma^{-1}(f(x))$

If $\underline{w} \in \mathbb{R}^d$, $b \in \mathbb{R}$, then with d+1 random queries of d-dim inputs, adversary can solve for $\underline{w}$ & b.

Label-only Attacks: If model only outputs $y = \text{sign}(f(x))$ pick 2 samples $\underline{x}_{-1}$ & $\underline{x}_1$ of different classes. Then by binary search on the line btw $\underline{x}_{-1}$ & $\underline{x}_1$ find $\underline{x}^*$ that lies on the boundary $f(x) \approx 0$. Given d+1 such points, we can recover $\underline{w}$ & b.

17.3 Stealing Deeper Models: for ReLU-NNs: Identify linear regions in model's input space & then solve for params that def. the linear fct.

17.4 Attacking Transformers: **Goal:** find hidden dim h

Transformer: $f: |V|^N \to [0, 1]^{|V|}$. prompt (of N tokens from vocab V) ↦ prob distr over V. ⇔ $f(p) = \text{softmax}(W \cdot g(p))$

$W \in \mathbb{R}^{|V| \times h}$ embed. proj. mat., $g: |V|^N \to \mathbb{R}^h$ rest of transf. Normally $|V| \gg h$.

Recovering h: (Attacker has access to logit·vec W·g) Logit-vec's lie in h-dim subspace bec last layer is a projection $\mathbb{R}^h \to \mathbb{R}^{|V|}$ ($|V| \gg h$)

Approach: 1) Query the model on n random prompts p_i & obtain logit vectors $W \cdot g(p_i)$. 2) Construct mat Q with each column being a logit vec. Note that $Q = W \cdot H$. 3) Return rank of Q

18 Privacy

18.1 Cryptographic Privacy

Secure training: Adversary only learns final model f & not $\mathcal{D}_{train}$

Secure Inference: Client calls ML model on Server. Client learns f(x) for input x & Server learns nothing.

Practice: large overhead (computation & commun.)

18.2 Encoded $\mathcal{D}$: **Idea:** encode $\mathcal{D}$: $E(x_i)$, send to central server, use encoded $\mathcal{D}$ to train model f.

2 cond: enc. should allow server to learn a useful mod. f but should not allow the server to learn "too much" about $\mathcal{D}$. **Issue:** impossible to simultaneously satisfy!

Instance Hiding: For img: **Idea:** combine private image x w/ public images $x_1, \ldots, x_B$: $x' := \lambda_0 x + \sum_{i=1}^B \lambda_i x_i$, where λ_j are random weights. **Obs.:** doesn't hurt training, but improves it. **Improvement:** scale each pixel in x' to [-1, 1] & randomly flip signs.

TxtHide: Similar, but in embed. space.

Federated Learning: Central server sends current model f to clients → compute grad.-update locally → server aggregates gradients → sends updated f back to client. ↯ encoding leaks info about $\mathcal{D}$.

Grad is deterministic fct of $\mathcal{D}$. → reconstructable

Idea: find $\tilde{x}_1, \ldots, \tilde{x}_B$ s.t.: $\|\sum_{i=1}^m \nabla f(\tilde{x}_i) - g\|$ is minim. for $g = \sum_{i=1}^B \nabla f(x_i)$ w/ private inputs $x_1, \ldots, x_B$.

18.3 Privacy Backdoors: Server sends malici. server updates to client s.t. $g_I \approx x_j$. ↯

Extension: host mal. public f that, when locally FT, sends $\mathcal{D}$ to host.

19 $\mathcal{D}$ Extraction:

Def: (Extractable Memo.) For a model f & an exmpl x from $\mathcal{D}_{train}$, we say that x is extractablly memorized if an adversary can construct prompt p s.t. $f(p) = x$.

19.1 Measuring Memorization: given $p_1, \ldots, p_n$ prompts of few tokens, evaluate $f(p_1), \ldots, f(p_n)$, check if $x_1, \ldots, x_n$ contain large substrings $\in \mathcal{D}_{train}$

Results: Base Models memorize >0%. Aligned models 0%, BUT(!): Possible to constr. p's s.t. Aligned models leak more than base models.

One can "reverse" the alignment by FT.

19.2 Heuristic Defenses: **Memorization Filters:** Sample $x \sim f(p)$ & check if $\text{substring}(x) \in \mathcal{D}_{train}$. (Bloom Filters). Or: memory free decoding: given p, obtain logits & sample tokens w/ rejection

Limitation: Only flags exact overlaps!

Downside: whenever model cannot replicate input p e.g. "ABC, ABC, ..." then "ABC" $\in \mathcal{D}_{train}$

Deduplication: $\mathcal{D}_{train}$ contain duplicates. Duplication causes memorization

20 Differential Privacy:

"It shouldn't matter if one individual was $\in \mathcal{D}_{train}$ or not."

Def: Two Databases D, D' are neighboring, D~D', if |D| = |D'| & they differ in ≤1 row.

Def: A randomized Algo $M: \mathcal{X}^n \to \mathcal{Y}$ is ε-diff. priv. if ∀ neighboring D~D' & ∀ S ⊆ $\mathcal{Y}$: $\mathbb{P}(M(D) \in S) \leq e^\varepsilon \mathbb{P}(M(D') \in S)$

Thm: Let $M: \mathcal{X}^n \to \mathcal{Y}$ be ε-diff. private & let $f: \mathcal{Y} \to \mathcal{Z}$ be any (possibly random.) fct. Then $f \circ M$ is ε-diff. priv.

Thm: Let $M = (M_1, \ldots, M_k)$ be a seq of Algo's s.t. M_i is ε_i-diff private & M_i can depend on outputs of $M_1, \ldots, M_{i-1}$. Then M is $(\sum_{i=1}^k \varepsilon_i)$-differentially private.

20.1 Laplace Mechanism:

Def: The ℓ_1-sensitivity of a fct $f: \mathcal{X}^n \to \mathbb{R}^k$ is def as: $\Delta_1 = \max_{D \sim D'}\|f(D) - f(D')\|_1 = \max_{D, D'}\sum_{i=1}^k |f(D)_i - f(D')_i|$

Def: Laplace: $f(x|\mu, b) = \exp(-\frac{|x - \mu|}{b}) / 2b$

Def: Let $f: \mathcal{X}^n \to \mathbb{R}^k$ be a fct w/ ℓ_1-sensitivity Δ_1 & let ε > 0. Then the laplace mechanism is def as: $M(D) = f(D) + (Y_1, \ldots, Y_k)$, $Y_i \overset{iid}{\sim} \text{Laplace}(0, \frac{\Delta_1}{\varepsilon})$.

Thm: The Laplace Mechanism is ε-DP.

Proof: $\frac{p_D(y)}{p_{D'}(y)} = \frac{\prod_{i=1}^k f(y_i | f(D)_i, \frac{\Delta}{\varepsilon})}{\prod_{i=1}^k f(y_i | f(D')_i, \frac{\Delta}{\varepsilon})} = \ldots = \leq \exp(\frac{\Delta_1}{b}) = \exp(\varepsilon)$

20.2 DP Learning: Algo (DP-SGD): $\theta_0 \leftarrow \$\Theta$, for t = 1:T {
Sample $S \subseteq [n]$ s.t. |S| = m, $g_i \leftarrow \nabla_{\theta_t}\ell(\theta_t, x_i, y_i) \ \forall i \in S$,
$g_i \leftarrow C \cdot g_i / \max(C, \|g_i\|_2) \ \forall i \in S$, $\bar{g} \leftarrow \frac{1}{m}\sum_{i=1}^m g_i$,
$\theta_{t+1} \leftarrow \theta_t - \eta(\bar{g} + \mathcal{N}(0, \sigma^2\mathbb{1}))$ }

Rem: · Subsampling amplifies privacy: $\frac{m\varepsilon}{n}$-DP
- Large & Good pretrained model ∝ free privacy
- What is the "right" granularity?
- Few-shot learning: perfect privacy. Problem: pretraining

21. Membership Inference:

Goal: $x \in \mathcal{D}_{train}$?

MIA yield a lower bound on the leakage.

Game b/w adversary & Challenger: A wins if guesses if $x \in \mathcal{D}'$.

Success Evaluation: Want to analyse worst-case leakage. TPR = $\mathbb{P}(b' = 1 | b = 1)$, FPR = $\mathbb{P}(b' = 1 | b = 0)$ $b = \mathbb{1}\{x \in \mathcal{D}\}$, $b' = \mathbb{1}\{\text{Adv. thinks } x \in \mathcal{D}\}$.

Thm: An Algo is ε-DP iff TPR/FPR ≤ e^ε.

Rem: Particularly interested in increasing TPR at low FPR

21.1 Designing Membership Inference Attacks.

$H_0: x \notin \mathcal{D}_{train}$, $H_1: x \in \mathcal{D}_{train}$ ⇒ 1) Select test statistic T. 2) Estim distr. of T(f, x) under H_0 & H_1. 3) Apply Likelihood ratio test. Let $T(f, x) = \mathcal{L}(f, x)$ loss fct.

Optimal test: $\frac{p(T(f,x)|H_1)}{p(T(f,x)|H_0)} > \tau$ (τ threshold)

Approach: Train N shadow models, half of which are trained w/ x & half w/o. Compute losses & model distr. as 2 Gaussians: $\mathcal{N}(\mu_{in}, \sigma_{in})$, $\mathcal{N}(\mu_{out}, \sigma_{out})$

↳ $\frac{p(\ell|H_1)}{p(\ell|H_0)} = \mathcal{N}(\mu_{in}, \sigma_{in}) / \mathcal{N}(\mu_{out}, \sigma_{out})$

Rem: MI only works on outliers / worst-case exmpls

21.2 Challenges for MI on LMs:

Choice of $\mathcal{L}$: $\mathcal{L}_{CE}(f, x) = \sum_{t=1}^T \log(f(x_t | \ldots x_{t-1}), x_t) \to \mathcal{N}(\mu, \sigma)$

Per Token: $\mathcal{L}_t = \log(f(x_1 | \ldots x_{t-1}), x_t) \to \mathcal{N}(\underline{\mu}, \underline{\Sigma})$ multivar

Estim $\mathbb{P}(\mathcal{L})$: shadowmodels unfeasible.
- Guess $x \in \mathcal{D}$ if: $\mathcal{L}(f, x)$ low even if x is not easily compressib.
- Compare $\mathcal{L}(f, x)$ w/ $\mathcal{L}(f, x')$ for $x' \in B(x, \delta)$ ⇒ If much lower loss → high prob in $\mathcal{D}$

Proving $x \in \mathcal{D}$: **Random Canary:** Add r.c. before data x ↳ $x' = (c, x)$ → If $x' \in \mathcal{D}$ than $\mathcal{L}(f, x') \ll \mathcal{L}(f, c' + x) \ \forall c' \in C \setminus \{c\}$

↳ Rank check **Data Extraction:** If f can recover missing compl. → $x \in \mathcal{D}$