ANLP

Representation -

- Words, tokens, vectors, embeddings used to be similar to words

Chomsky view - computational linguistics
Shannon view - information theoretic ("encoding" & "decoding")

Splitting tokens used to be done at word/sentence level (look for space/period..)
Now, they are subword

Early Representations - one hot vector, tf idf, cooccurrence matrix with SVD
Limitations - Sparsity, ambiguity, lack of context

Deep Learning in NLP:
Word 2 Vec
(CBOW)
↳ Continuous Bag of Words: Predict middle word from context
↳ Skip Gram (SG): Predict context from middle word

"Word Arithmetic" is now possible
E(King) - E(Man) + E(Woman) ≈ E(Queen)

Limitations: Ambiguity, Context (to a lesser extent)

Modelling

Language Modelling Problem:
Probability of a sequence (predict next word)
Before: Use conditional probability, with finite context (for practical reasons)

N-Gram Language Mode:
Sparse, doesn't exploit word similarity, finite context

Neural Network Language Modelling
Sparsity - solved, word similarity - solved
context - not solved, computational complexity

Recurrent Neural Networks

Used for time series data (stocks, weather...) & many NLP tasks.
Sequence to sequence (translation, speech recognition), classification tasks
Limitations: Long distance dependencies, vanishing & exploding gradients

LSTM - solved vanishing gradients but not long distance dependencies

2018, ELMo : Embeddings from Language Models Pre-Trained biLSTM for contextual embedding

Limitation of LSTM : Computationally complex (softmax), slow (ish), needs labelled data, transfer learning not possible

Attention:

Focus to individual components of sentences, interactions of words with other words

BERT : Bidirectional Encoder Representations from Transformers Pre-trained transformer encoder for sentence embedding → parallelization
↳ masked language modelling task

Today:

Representation : Embeddings from LLMs

Modelling : Encoder only : BERT, XML
Encoder - Decoder : T5, BART
Decoder only : GPT, LLaMA

NLP

1950s - 2010s : Symbolic → usually take in some input of symbols to do some task like classification.

Symbolic AI cares only about tokens presented as input & extracts relations & features

Ways to break text into elements:

- characters
- words (boundary: space)
- boundary: punctuation
- boundary: conjunctions

Phrasal vs discoursal boundaries matter. Ex: Ram and Shyam went out.
Cant split on this "and".

In images, two pixels are not related by themselves, but due to the real world entities they represent. However, in text, the grammar enforces a structure which gives inherent relations between elements.

In English, it is always actor verb object
Ex: The dog bit the man, but in many other languages, it can be actor object verb or object actor verb depending on the modifier.

Much more variation in text & language compared to images.
Some languages have no spaces
Ex: Mandarin, Thai

Language ≠ Text
Text loses some non verbal information like emphasis, intonation

"Translation is NLP-Complete" → needs all NLP tasks

① Words are complex - morph & encode some properties - Morphology
② Assign parts of speech
③ How are POS arranged? : Syntax (Depends on language)

Some languages are low morphology (Ex: Hindi), high morphology (Ex: Kannada). Hard to translate from low to high morphology languages

Noun Phrase

Specifier	Adjective	Noun
the	blue	house

English <u>requires</u> a subject. In stative sentences, a subject is artificially added. Ex: It's raining.

Pattern based learning ≠ Language Understanding

ANLP 10/08/2024

Representation of "Meaning" in Text

"Universality of language" - Part of human brain design
is an argument

Diff. languages have diff. devices to achieve the same task

Language $\xleftrightarrow{\text{informs}}$ Writing

Morphemes - parts of a word Ex: friend(s)
In some languages, they are not attached at the end

Suffixes can change the part of speech of a verb.
(in general, affixation
(prefixes, infixes))
↳ Ex: Arabic

Structure is available at different levels of the language
↳ sentence,

Creole can be at an individual level as well

Dialect codevelops with a language & one is chosen as a standard (Hindi is not a dialect.)

MLE: $\underset{\theta}{\text{argmax}}\ P(D; \theta)$
↳ data ↳ set of parameters

Statistical Language Modelling

Grammar is learned by proxy by learning high frequency phrases

WordNet - a data structure that contains meanings of groups of words

ways to represent the words - index of word in dictionary, one-hot vector

---

After abstracting information out, we have a numerical representation → model works on this representation → abstraction depends on the domain

Meaning is defined by <u>context</u> + consensus

Distributional Semantics - Meaning of word is distributed across its content.

~~SVD~~ SVD: $A = UWV^T$

Linguistics

→ Syntax

→ Semantics → lexical - evolves quickly
→ X-compositional (Idiomatic / pseudo id. non ...)

→ Discourse → References
→ ~~Causality~~ Relations
→ Pragmatics (a social construct)

Statistical Understanding of Language

Model : P(D | θ)

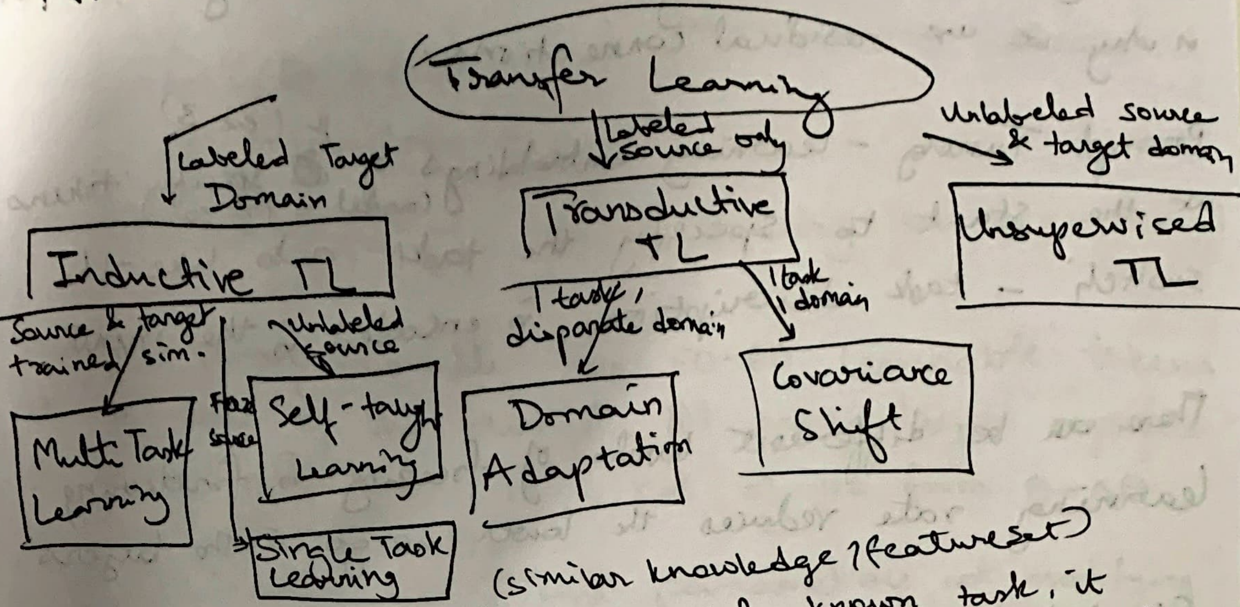
H: entropy , 2^H : perplexity (of the model) lower is better

Words appear together because of syntax & semantics.

Proxy task to learning meaning - word prediction

CS224n

ANLP



To perform a "subset" (similar knowledge / feature set) of a models known task, it ~~&~~ usually suffices to just use this model. Ex: vgg for large scale ~~cla~~ image classification can be used for image segmentation. This is inductive single task learning (freeze original model & train the secondary model on the target domain). Here, classification is used as a proxy task.

Unsupervised TL is hard to achieve, but as an ex: using an LLM for sentiment classification → this depends on whether the LLM has seen definition & examples of sentiment in its training. ML in general is a function approximator ~~b/w~~ a domain & a range, but we have limited the domain & range through training data. Also, we need architecture, features & outputs affects how well we can approximate the fn.

A neural network is like a multitude of models learning stuff based on different switches → hard part is separating them out. In NTP, the switch is the context → the task is to predict next token in this context. For the prev. ex., "Predict the sentiment" given as a prompt is the switch. The model is learning the switches & the fn. approximation for this switched task → the task is encoded.

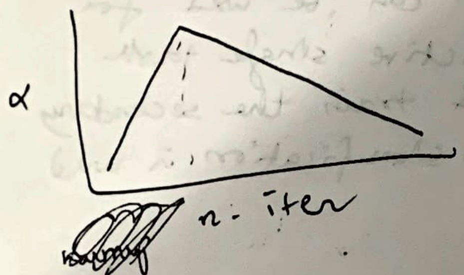
In a deep network, only the last few layers are task dep., the others learn general abstractions & the error does not propagate all the way to the initial layers. Which is why we use residual connections.

Prompt Tuning - learning embeddings of k (ex 3) specific tokens (model is frozen) at the start to specify the task - acts like the switch - task description is encoded in the input

There can be different levels of freezing in finetuning. learning rate reduces the lower we go in the layers.

Slanted triangular learning rate

To prevent catastrophic forgetting.



Problem:
Too many parameters to finetune in Llama & few hundred k samples are not enough.

14/9/24

- When fine tuning / transfer learning, we (usually) dont need ALL of what the model has already learnt.
- Look at it as the info. content needed to solve the og task > info content needed to solve the smaller task. (Subset of info content & not task itself)
- Feature representation for an input is an abstraction of the input, (at every layer). Features can be engineered or learnt, they may not always be named.
- Model → Architecture A + Parameters Θ & is
  $f : x \to y$ where $f(y; x, A, \Theta)$

- $\hat{\theta}$ ~~→ transform (θ, $\Delta\theta$)~~

↳ almost always addition

$\hat{\theta} = \theta + \Delta\theta$

$\Delta\theta = \text{transform}(\theta)$

- ~~Model editing is~~ Model editing : ex bias removal

- Prefix tuning usually has 50-60 learnable tokens at the start of the prompt. At every layer a prefix is being attached. It is not of the form $\theta + \Delta\theta$, but of form $\theta + \underbrace{\theta_p}$ Extra parameters, not modifying existing ones

- Prompt tuning - dont change model, change vocabulary to add extra tokens. Not very effective - lot of change required.

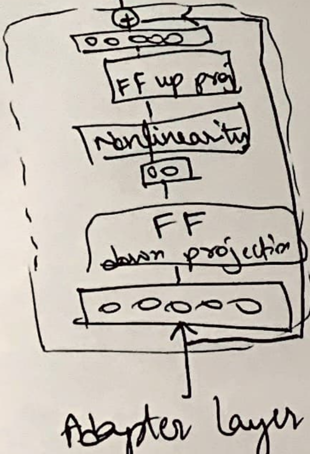
Trying to change at the source (prompt tuning) vs changing at every layer (prefix tuning)

less effective but less params — more eff. but more params

Convergence is slow

Adapters:

Add at every transformer layer b/w the 2x FFN & the layer norm. Viewing the problem as sieving a river instead of moving a river → compression instead of steering



At the FF down projection, the update (gradient) for a feature depends on its importance (directly proportional)

Training adapters is fast but
Problem : We're adding layers - more compute, memory, slower inference. Also adapter layers have to be processed sequentially at inf. time.

Re-parameterizing the model into something ~~[illegible]~~ easier to train : Ex : LoRA, (IA)^3
(more efficient)

Weight matrices are usually full rank - all rows/columns are important, aka. "well distributed"
For a simpler task, only a subset of this info is needed ; if we could get the rank for this subset, ~~[illegible]~~ - it would be low rank