Advanced Methods in Text Analytics

NLP Applications

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

The Story So Far

- **This course** was focused on **the latest *methods* for NLP**
  - Basic ML and DL (fundamental for understanding NLP *methods*)
  - Recap of basic NLP (e.g. n-gram language models)
  - FNNs for language modeling
  - Static Word Embeddings (specifically, the skip-gram approach)
  - RNNs for language modeling
  - The Transformer Architecture
  - Transfer Learning (pre-training + fine-tuning)
  - Tokenization
  - Large Language Models (LLMs)
  - Multilingual NLP (next lecture)
- In this lecture: **common applications of these methods**
  - Specifically, **how LLMs are evaluated** on specific NLP tasks, e.g. natural language inference (NLI), natural language understanding (NLU), etc.
  - **Real-world NLP applications**, e.g. machine translation, dialogue systems

Outline

1. LLM Evaluation

2. Real-World Applications

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

LLM Evaluation

Why the focus on LLMs?

- In this section: **methods for evaluating LLMs**
  - But why LLMs, specifically?
- Recall LLM lecture:
  - **Most NLP tasks today solved with pre-trained language models (LMs)**
  - **Pre-trained LMs:** models like BERT (2018), or LLMs like LlaMA-3 (2023)
  - How to use BERT? Already covered in Transfer Learning lecture
- But how to evaluate/use LLMs still unclear/challenging in most settings
  - Why?
- **Virtually all LLMs are causal language models (CLMs)**
  - Recall **CLMs:** LM trained to predict next word in input token sequence
  - But **many NLP tasks** are evaluated as **classification tasks**
  - E.g. sentiment analysis is usually the task of predicting one of three labels: "positive", "negative" or "neutral"
- Important question: **how to use a CLM for classification tasks?**
  - Henceforth, acronyms LLM and CLMs used interchangeably

LMs as Functions

- At the core, a **LM** is a (learned) **function *f: T --> P***
  - ***T*** is set of all possible sequences of words (i.e. tokens) from vocabulary *V* of one (or more) natural languages, e.g. English, German, etc.
  - ***P*** is probability simplex of size *|V|*
  - In other words, *p = f(t)* for *t ∈ T* and *p ∈ P* provides a probability distribution over *V*, i.e. LLMs output a probability value for every word in *V*
- Visually, for any *t ∈ T* with *T = {a, b, … abalon, … , elephant, … , Ente...}*, *f(t)* provides such a distribution:

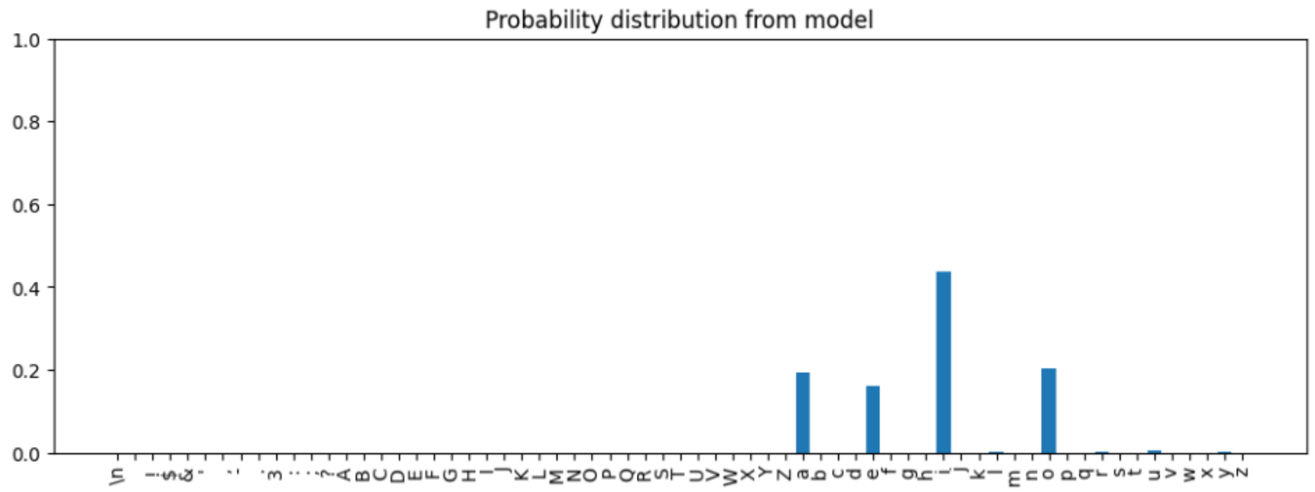


Image source

Interpreting LLM Outputs

- Function ***f*** **does not say anything about:**
  - **1. How to interpret** the distributions in ***P***
  - **2. How to use** the distributions in ***P***
- **1. How do we typically *interpret*** a distribution ***p*** $\in$ ***P*** given by LLMs?
  - $p_i = f(t)_i$ is *conditional* probability that *i-th* word in *V is next* given $t \in T$
- The **training process of CLMs allows us to interpret *p***
  - Models "forced" to adjust parameters so *next words* get high probabilities
- **Different training objective --> different interpretation of *p***
  - E.g. *p* in masked language models (MLMs), $p_i = f(t)_i$ is conditional probability *i-th* word is "surrounded by" given $t \in T$ (recall that we don't use MLMs like that, we use them for their contextualized representations)
  - Similarly, we could train an "inverse CLM" if we construct the corresponding training examples using self-supervision
- **Interpreting *p*** is about **doing *inference* with the model**
  - One *use* of *p* would be for making predictions (inference != prediction)

Using LLM Outputs

- **2. How do we typically *use*** a distribution ***p*** $\in$ ***P*** given by LLMs?
  - Most common application: **autoregressive text generation**
  - Recall autoregressive generation: each word in generated sequence of words is predicted *conditioned on previous words* in sequence
  - It's a natural application given the training process in CLMs
- How do we use ***p* for autoregressive text generation**?
  - Given initial prompt *t* (starting sequence of words), *sample* next word *w* from output distribution *p* given by LLM (e.g. using beam search)
  - Set $t = t + w$, then repeat step above until stopping criterion is satisfied
  - Different sampling methods (e.g. top-k) show *we can* use *p* differently
- Note we can use *p* for other applications
  - E.g. compute probability of some $t \in T$ using the chain rule
  - That is, $p(t) = p(t_1|prompt)p(t_2|prompt,t_1)\ldots p(t_n|prompt,t_1,\ldots,t_{n-1})$
- Important question revisited: **how to use *p* for classification tasks?**
  - Let's see how GPT-3 did it (arguably the first LLM)

Evaluating GPT-3 For Classification

- GPT-3 showed increasingly more ICL ability as model got larger
  - ICL (in-context learning): **given prompt** with **text instructions** for a task and/or corresponding **examples of the task**, LLM is able to "**solve"** the **task without weight updates**
- The model was evaluated on two kinds of tasks
  - **Free-form completion:** output is any text, e.g. translation (discussed later)
  - **Multiple choice completion:** only limited set of output tokens are meaningful, e.g. the set of labels in classification tasks (discussed now)
- Example of (a very general) classification task: LAMBADA dataset
  - Task: complete last word in given sequence, i.e. language modeling task
  - Correct answer "far" from missing slot, it tests **long-range dependencies**
- How was this task done with ICL? Example from GPT-3 authors:
  - "Alice was friends with Bob. Alice went to visit her friend ____. --> Bob.
  - George bought some baseball equipment, a ball, a glove and a ____. --> "
- **Which sampling method would be suitable for prediction?**

Per-Token Likelihood (1)

- **GPT-3** was **evaluated** on **classification tasks** using **greedy sampling**
  - Specifically, they would **compare log-probability of all tokens in each label**
  - Note that even a single label may take more than one token
  - E.g. word "positive" may use single token, whereas word "neutral" may use two: "neu" and "tral" (imagine "positive" more common than "neutral")
- If we **compare log-probabilities across labels** with different numbers of tokens, we **may be biased by token length** of some labels
  - Thus, GPT-3 evaluation **normalized log-probabilities by number of tokens**
- Concretely, the **normalized *prediction score*** for prediction sequence $t_{m:n}$ given input sequence $t_{1:m-1}$ is given by:
  - $s(t_{m:n}) = (1 / (n-m)) * \Sigma_{i=m}^{n} \log p(t_i | t_{1:i-1})$
  - That is, these are the probabilities from output distribution *p* given by a CLM, but normalized by length (in number of tokens) of output sequence
- Let's visualize this!

Per-Token Likelihood (2)

- Say LLM-1 that tokenizes **two labels** as follows
  - Label "positive" in 1 token, label "neutral" in 2 tokens: "neu" and "tral"
- Imagine the following **unnormalized prediction scores** for each label
  - That is, $s(t_{m:n}) = \Sigma_{i=m}^{n} \log p(t_i | t_{1:i-1})$
  - P_{LLM-1}("positive") = log p("positive"|<s>) = 0.6
  - P_{LLM-1}("neutral") = log p("neu"|<s>) + log p("tral"|"neu") = 0.3 + 0.5 = 0.8
  - P_{LLM-1}("neutral") > P_{LLM-1}("positive") so **we'd pick label "neutral"**
  - But note **"neutral" has higher score due to its number of tokens**
- Now let's **normalize those scores by token length**
  - That is, $s(t_{m:n}) = (1 / (n-m)) * \Sigma_{i=m}^{n} \log p(t_i | t_{1:i-1})$
  - P_{LLM-1}("positive") = log p("positive"|<s>) = 0.6
  - P_{LLM-1}("neutral") = log p("neu"|<s>) + log p("tral"|"neu") = (0.3+0.5)/2 = 0.4
  - Now **we'd pick label "positive"**
- Thus, **normalizing log-probabilities by token-length is crucial!**

Per-Character Likelihood

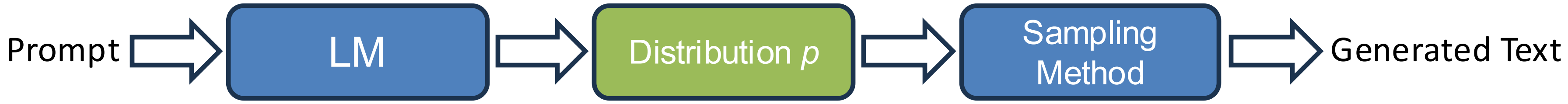
- **Different tokenizers may encode a given input sequence in a different numbers of tokens**
  - E.g. word "namaste" in LlaMA-3 uses 3 tokens
  - Another LLM specialized on Hindi might use a single token
- In this case, we can **normalize scores by number of bytes**
  - E.g. $s(t_{m:n}) = \Sigma_{i=m}^{n} \log p(t_i | t_{1:i-1}) / \Sigma_{i=m}^{n} B_i$ where B_i is number of bytes in token t_i, i.e. we normalize by length in bytes, not in tokens
  - **Recall decoding with BPE-style tokenizers:** start from bottom characters, then merge in order of merges found during tokenizer training
  - A new token that comes from merging two *n*-bytes token takes *2n* bytes
  - Thus, only sensitive to spelling/vocabulary (i.e. still a problem when comparing across languages)
- Note: it's **not common to compare log-probabilities across models**
  - We can't assume LLMs are calibrated
  - But may be necessary to compare across languages (more in next lecture)

From Prediction Scores to Metrics

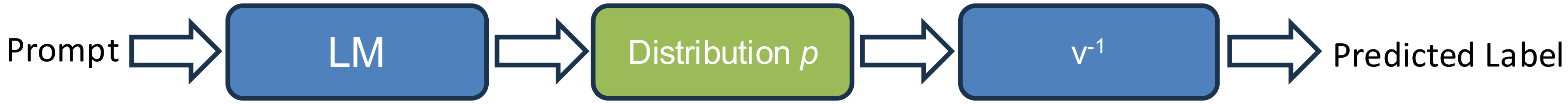
- Whether unnormalized or not, prediction scores are aggregations of the probabilities of tokens in the generated answer
- **Given those probabilities**, we can **use them as with any classifier** to compute performance metrics
  - For example, for **classification tasks**, we can **pick label $l \in L$ with highest probability for each example x**, where **L is set of labels** for given task
  - Then, given pairs of inputs and outputs (x,l), we can compute metrics like accuracy or F1-Score.
  - **More recently**, as models got better, it's often **enough to do greedy sampling**, as models almost always predict one possible label.
  - I.e. we don’t restrict ourselves to L, but instead compare probabilities across the entire vocabulary
- **Reminder:** we can use output distribution however we want
  - For example, authors of GPT-3 found benefit **on some tasks** by computing probabilities of output sequences when removing some tokens used for formatting, e.g. "Answer: "

Verbalizers (1)

- Sampling methods can be changed for different text generation tasks
  - E.g. greedy for factual-based question answering, top-p for creative tasks



- Similarly, **we can use LM output in different ways to make predictions**
- The process of **mapping a string to a label** in a classification task is often **formalized with verbalizers** (Schick et al. 2021)
- **Verbalizer:** *injective* function *v: L --> V* that maps labels from some set of labels *L* to language model vocabulary *V*
  - **Injective:** each element in domain mapped to *distinct* element in range, i.e. no two elements in domain get mapped to same element in range



Verbalizers (2)

- Some **examples of verbalizers** for a **binary classification** task can be:
  - Label "True" --> Token "True", label "False" --> Token "False"
  - Label "True" --> Token "1", label "False" --> Token "0"
  - Label "Positive" --> Token "pos", label "Negative" --> Token "neg"
- The **choice of verbalizer must be specified** to the model
  - Can be done via examples in ICL demonstrations
  - Or by specifying the expected labels in instructions in a zero-shot setting
- Again, the **choice of verbalize can make a difference** on model performance
  - Can be especially evident in weaker models that may prefer natural sounding labels over numerical ones
- So far: how to use output of LLM for classification tasks
  - What about "generative" or "free-form" tasks?
  - E.g. summarization or translation.
  - This brings us to the core of LLM evaluation

Challenges in LM Evaluation

- Key challenge according to Biderman et al. (2024): **language ambiguity**
  - Many syntactically different ways to express *semantically equivalent* ideas
  - Best tools to say if two sentences are semantically equivalent? LLMs!
  - I.e. we can't have a model evaluate itself
- How about **human effort**?
  - Humans can always manually evaluate the output of models
  - But this is **expensive**, evaluation data has hundreds/thousands of examples
  - And humans can certainly also make mistakes
  - Instead, human effort **usually goes to creation of gold answers** in benchmark construction
- Given gold answers, we can compute **metrics** automatically
  - E.g. ROUGE score compute lexical overlap with gold answer
  - BLEU is a score designed for machine translation
  - Exact-match: proportion of answers that exactly matches gold answer
  - Note that metrics can be flawed, e.g. BLUE has issues, ROUGE has issues

Model-Based Metrics (1)

- **Idea:** replace human effort with LLM effort
- **Motivation 1:** benchmark performance scores often miss nuance
  - Instruction-tuned vs non-instruction tuned models often perform the same
  - But humans known to prefer answers given by aligned models
- **Motivation 2:** ChatGPT shown to outperform humans as annotator
  - Gilardi et al. (2023) found ChatGPT to be better than crowd workers
  - Huang et al. (2023) found ChatGPT great at picking nuanced hate speech
- LLMs-as-a-Judge: ask model to evaluate generated answer
  - Judge model often (but not necessarily) different from evaluated model
  - Judge model must be strong LLM, very common choice: GPT-4
  - Open source alternative proposed recently: Prometheus
- **How to use an LLM as an evaluator?** Again, freedom.
  - **Pairwise comparison:** model chooses from given pair of answers
  - **Single answer grading:** model provides score to answer (give ICL examples)
  - **Reference-guided grading:** model compares answer to gold answer

Model-Based Metrics (2)

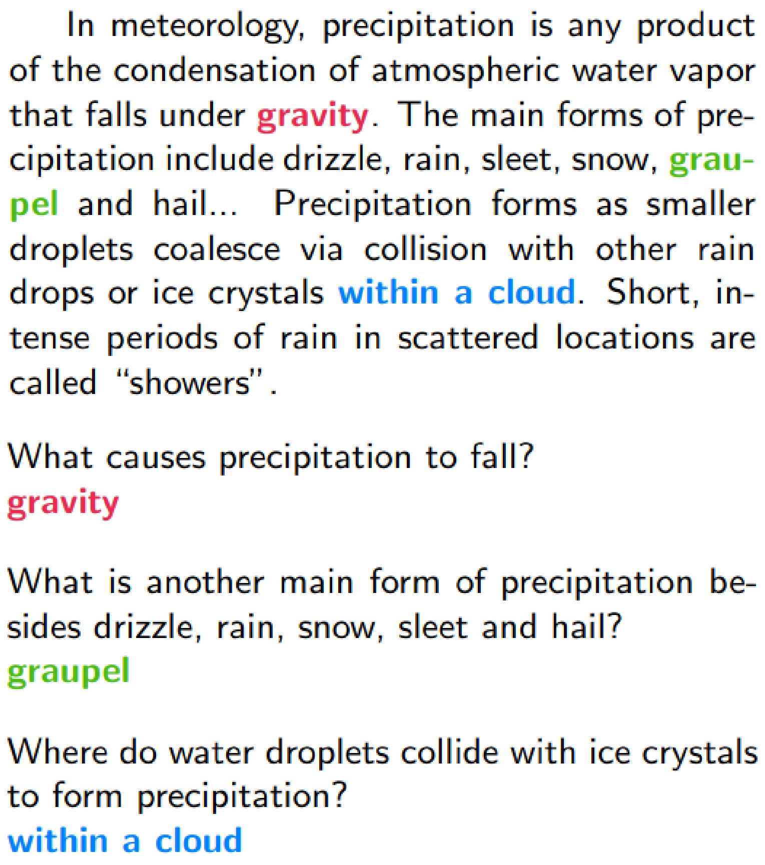
- Important 1: **reading generally easier than writing** for LLMs
  - I.e. evaluating text **not the same task** as generating text
  - Thus, even same model may evaluate its own answers given right prompt
  - This is what self-alignment is based on
- Important 2: **LLM-as-a-judge useful on settings without gold answers**
  - Model-based metrics when reference answers are available not new
- Example: BERTScore
  - Simple idea: given reference answer, compute token-level similarity with tokens in generated answer
  - They use greedy matching to match each generated token to most similar token in gold answer
- **Metrics only one component** of LLM evaluation
  - Another key component: **benchmarks**

Evaluation Benchmarks

- **Benchmarks:** standard datasets used by academia and industry to test model performance on some task
  - They are **essential for progress in academia**
  - But **test data must be treated as unseen** (honor code)
- **Too much focus on benchmarks can also be problematic**
  - Performance on benchmark may not translate well to real-world settings
  - After some years, researchers may inevitably overfit to test data by using known tricks from prior work
  - In the era of LLMs, difficult to know if some models may not have seen test data from benchmarks during training (data contamination)
  - Common to find flaws in widely used benchmarks after some years
- See nice overview of importance and desiderata of benchmarks here
- **Hundreds of benchmarks** have been **released over the years**
  - We go over just a few in the following slides
  - Some classics, some commonly used for evaluating LLMs

Question Answering

- The SQuAD dataset (Rajpurkar et al. 2016)
- **Task:** question answering to test reading comprehension
- **Questions:**
  - About Wikipedia articles
- **Answers:**
  - Segment (span) of text in article
- **Supervised data point:**
  - (paragraph, question, answer)
- **Evaluation metrics:**
  - Exact matching of answers
  - Proportion overlap with answer
- **Multilingual version:**
  - XQUAD (Artetxe et al. 2020)
  - SQuAD, professionally translated



In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

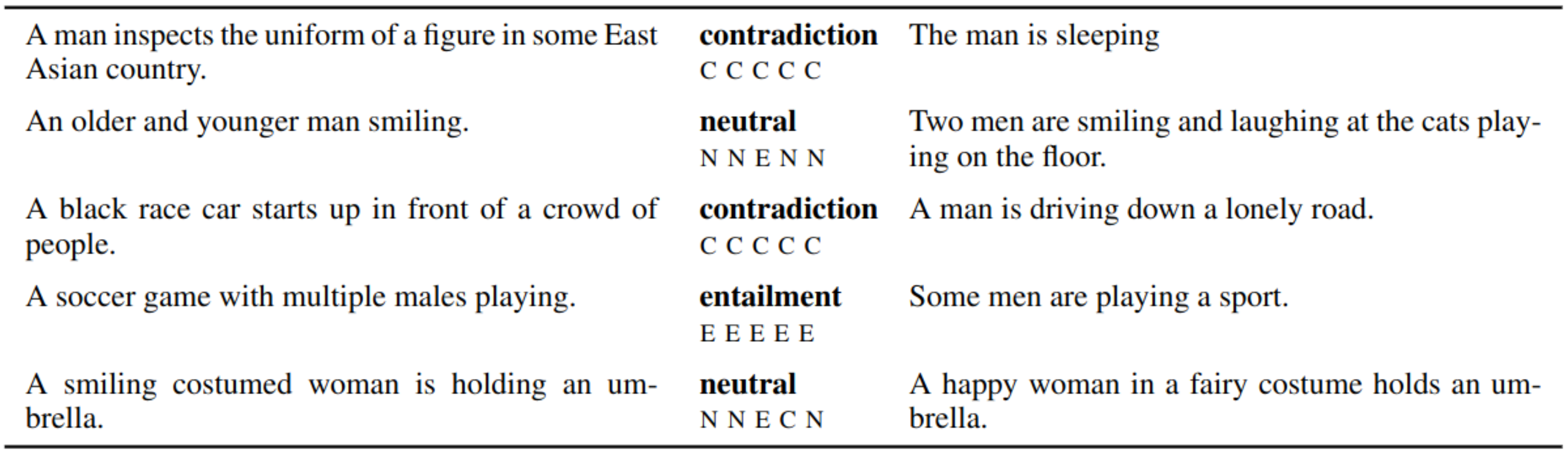
What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

Natural Language Inference

- The NLI dataset (Bowman et al. 2015)
- **Task:** natural language inference
- **Supervised data point:** (sentence 1, sentence 2, label)
  - Set of labels *L = {entailment, contradiction, neutral}*
- **Evaluation metric:** accuracy

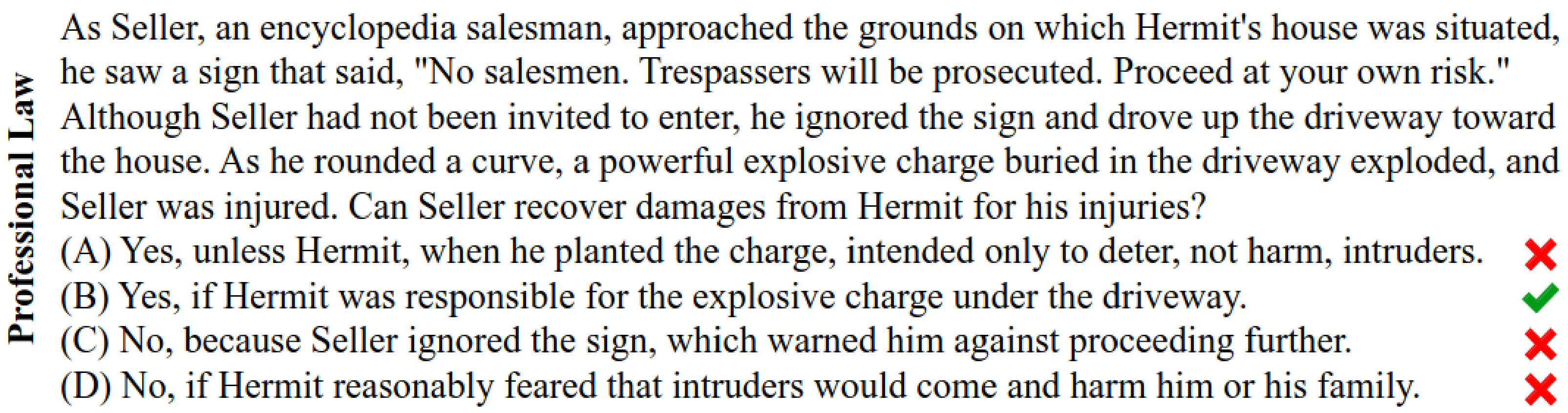


A man inspects the uniform of a figure in some East Asian country.	**contradiction** C C C C C	The man is sleeping
An older and younger man smiling.	**neutral** N N E N N	Two men are smiling and laughing at the cats playing on the floor.
A black race car starts up in front of a crowd of people.	**contradiction** C C C C C	A man is driving down a lonely road.
A soccer game with multiple males playing.	**entailment** E E E E E	Some men are playing a sport.
A smiling costumed woman is holding an umbrella.	**neutral** N N E C N	A happy woman in a fairy costume holds an umbrella.

- **Multilingual version:** XNLI (Conneau et al. 2018)
  - Extended NLI datasets to 15 languages, including low-resource ones

Language Understanding

- The MMLU dataset (Hendrycks et al. 2021)
- **Task:** multitask language understanding
  - **Subjects:** elementary mathematics, history, compute science, law, etc.
- **Supervised data point:** (multiple-choice question, correct answer)
- **Evaluation metric:** accuracy



Professional Law

As Seller, an encyclopedia salesman, approached the grounds on which Hermit's house was situated, he saw a sign that said, "No salesmen. Trespassers will be prosecuted. Proceed at your own risk." Although Seller had not been invited to enter, he ignored the sign and drove up the driveway toward the house. As he rounded a curve, a powerful explosive charge buried in the driveway exploded, and Seller was injured. Can Seller recover damages from Hermit for his injuries?

(A) Yes, unless Hermit, when he planted the charge, intended only to deter, not harm, intruders. ✗

(B) Yes, if Hermit was responsible for the explosive charge under the driveway. ✓

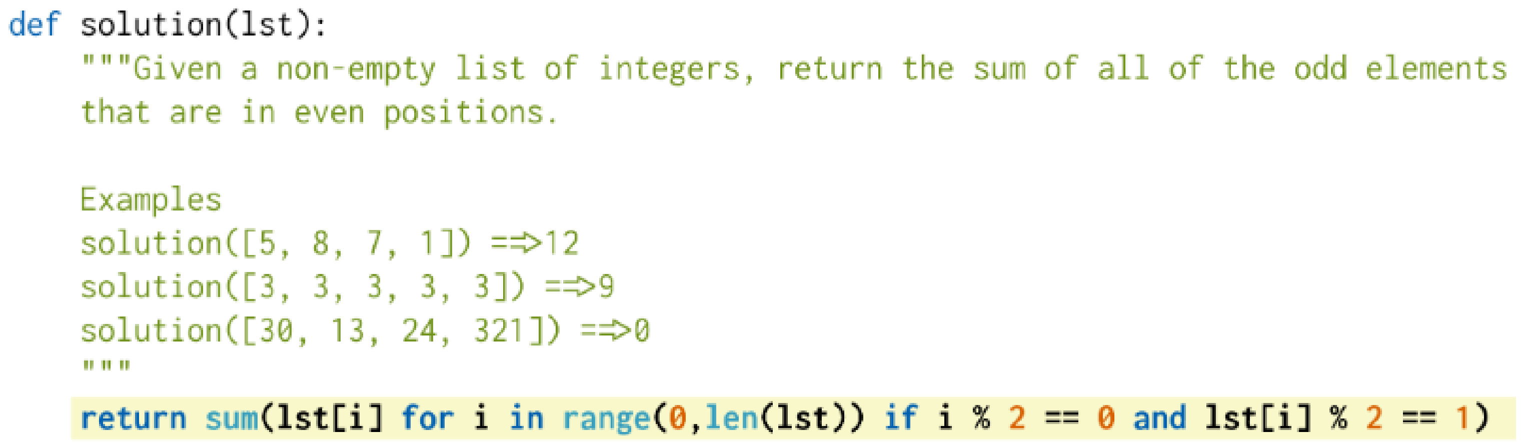
(C) No, because Seller ignored the sign, which warned him against proceeding further. ✗

(D) No, if Hermit reasonably feared that intruders would come and harm him or his family. ✗

- This dataset is **commonly used to evaluate LLMs**
  - Model performance on it is even commonly reported in the news

Code Generation

- The Human Eval dataset (Chen et al. 2021)
- **Task:** code generation
    - Specifically, generate stand-alone Python functions from docstrings
- **Supervised data point:** (function definition, unit tests)
    - **Function definition:** signature and docstring
- **Evaluation metrics:** pass@k
    - Generate *k* solutions, say problem is solved if 1 passes unit tests

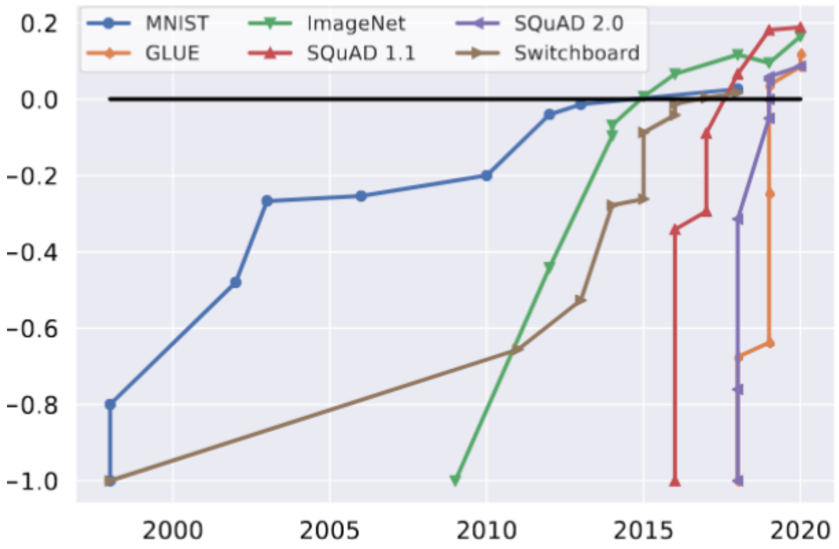


```python
def solution(lst):
    """Given a non-empty list of integers, return the sum of all of the odd elements
    that are in even positions.

    Examples
    solution([5, 8, 7, 1]) ==>12
    solution([3, 3, 3, 3, 3]) ==>9
    solution([30, 13, 24, 321]) ==>0
    """
    return sum(lst[i] for i in range(0,len(lst)) if i % 2 == 0 and lst[i] % 2 == 1)
```

Issues with Benchmarks (1)

- Benchmarks essential to scientific progress, but they do have issues
- **Validity**: how well benchmarks assess performance on a task
  - Subramonian et al. (2023) found consistent disagreement among NLP researchers w.r.t. definitions of tasks, translation of tasks to benchmarks
  - In other words, tasks and benchmarks are not so clear cut
  - Sometimes it's not clear what the task really is
  - Sometimes it's not clear how benchmark really measures the task
- **Saturation**: when models exceed humans in performance
  - Natural when using benchmarks
  - Lately saturation reached faster
  - Does not mean task is solved
  - Usually harder versions follow

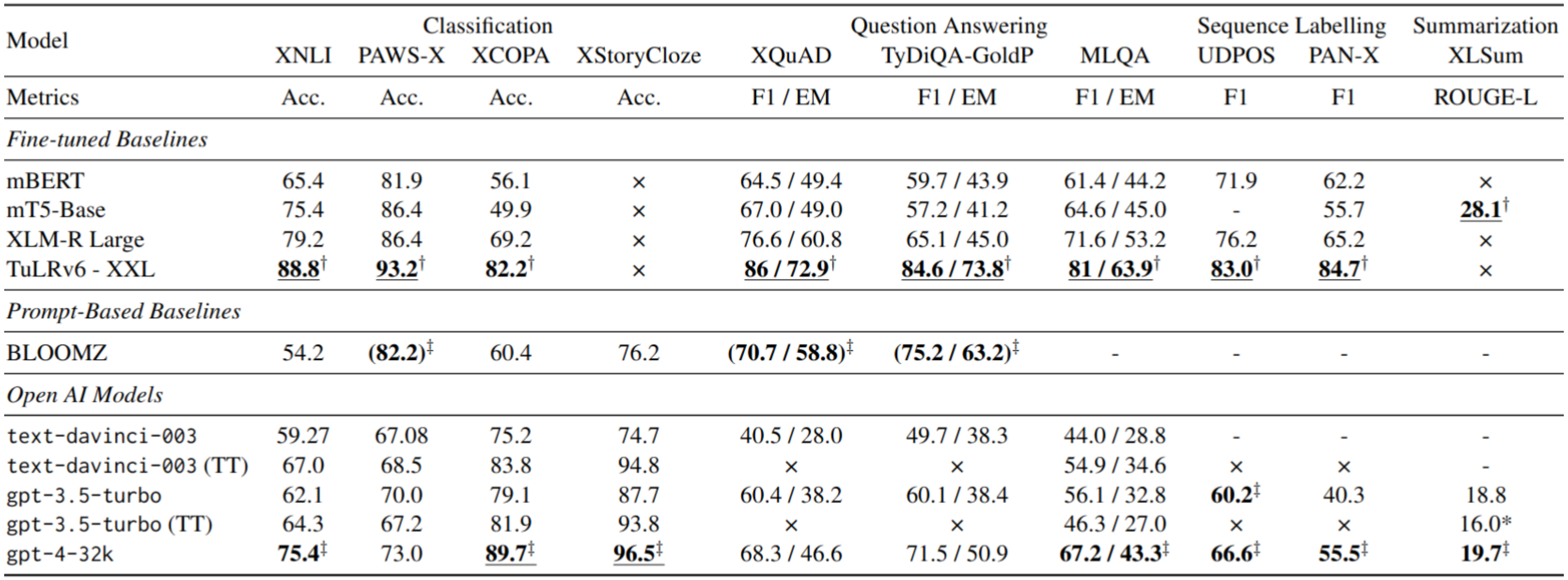


Issues with Benchmarks (2)

- Important concern with LLMs: **benchmark leakage**
  - LLMs train on vast amounts of data scraped from the internet
  - Benchmarks are publicly available online
  - This includes test data (remember the honor code)
- **Data contamination:** evaluating on examples included in training data
  - Evaluation examples need not be included explicitly in training data
  - But it's safe to assume this helps models perform better on benchmark
  - Roberts et al. (2023) found that GPT-4 performs better on benchmarks released before its cut-off date
  - Similar findings were reported by Li et al. (2023)
- **Data memorization**: when models reproduce benchmarks verbatim
  - Xu et al. (2024) used a method based on PPL and n-gram overlap to study how much can a model reproduce instances of public benchmarks
  - They found many models to be able to do that, e.g. Aquila and Qwen
  - But many models seem to be cleaner, e.g. Llama, DeepSeekMath

PLMs vs LLMs

- Fair question: **why** use **CLM (seq2seq) for** (discrete) **classification tasks?**
  - Why not something more designed for that, e.g. a MLM?
- Ahuja et al. (2023) report fine-tuned PLMs better than LLMs
  - Columns: tasks, underlined results: best overall, PLMs almost always better



Model	Classification				Question Answering			Sequence Labelling		Summarization
	XNLI	PAWS-X	XCOPA	XStoryCloze	XQuAD	TyDiQA-GoldP	MLQA	UDPOS	PAN-X	XLSum
Metrics	Acc.	Acc.	Acc.	Acc.	F1 / EM	F1 / EM	F1 / EM	F1	F1	ROUGE-L
Fine-tuned Baselines										
mBERT	65.4	81.9	56.1	×	64.5 / 49.4	59.7 / 43.9	61.4 / 44.2	71.9	62.2	×
mT5-Base	75.4	86.4	49.9	×	67.0 / 49.0	57.2 / 41.2	64.6 / 45.0	-	55.7	**28.1**†
XLM-R Large	79.2	86.4	69.2	×	76.6 / 60.8	65.1 / 45.0	71.6 / 53.2	76.2	65.2	×
TuLRv6 - XXL	**88.8**†	**93.2**†	**82.2**†	×	**86 / 72.9**†	**84.6 / 73.8**†	**81 / 63.9**†	**83.0**†	**84.7**†	×
Prompt-Based Baselines										
BLOOMZ	54.2	**(82.2)**‡	60.4	76.2	**(70.7 / 58.8)**‡	**(75.2 / 63.2)**‡	-	-	-	-
Open AI Models										
`text-davinci-003`	59.27	67.08	75.2	74.7	40.5 / 28.0	49.7 / 38.3	44.0 / 28.8	-	-	-
`text-davinci-003` (TT)	67.0	68.5	83.8	94.8	×	×	54.9 / 34.6	×	×	-
`gpt-3.5-turbo`	62.1	70.0	79.1	87.7	60.4 / 38.2	60.1 / 38.4	56.1 / 32.8	**60.2**‡	40.3	18.8
`gpt-3.5-turbo` (TT)	64.3	67.2	81.9	93.8	×	×	46.3 / 27.0	×	×	16.0*
`gpt-4-32k`	**75.4**‡	73.0	**89.7**‡	**96.5**‡	68.3 / 46.6	71.5 / 50.9	**67.2 / 43.3**‡	**66.6**‡	**55.5**‡	**19.7**‡

- Gap may have decreased since then, but question still fair

Summary

- Model evaluation is:
  - **Essential** part of ML/DL/LLM pipeline
  - **Designed** to test particular skill, performance on some task
  - **Interpreted**, as its results usually imply something about the model
  - **Challenging**, benchmarks/metrics can be flawed
  - **Requires** as **much care** as other parts of the pipeline: data, model, training
- To paraphrase George Box: **all evaluations can be flawed, but most are useful**
  - Thus, engaging with model evaluation **always requires critical thinking**
  - What do the results mean?
  - Why do they look like that?
  - Can model architecture explain them?
  - Can data explain them?
  - Can metrics explain them?
  - Can analysis of individual errors explain them?

Real-World Applications

The Machine Translation Task

- Usually abbreviated as **MT**
- Translation generally complex
  - Translating novels, poetry, etc. has nuance, artistry
- **MT use cases** more grounded
  - **Information access:** translate websites to local language
  - **Computer-aided translation (CAT):** start with MT-generated draft
  - **On-the-fly speech translation:** real-time translation, recent application
- Some aspects of language are universal across multiple languages
  - E.g. words for referring to people or common actions, e.g. eating, sleeping
- But **many challenges due to language diversity**, e.g.
  - **Word order:** *"He wrote a letter to a friend"*
    - *Tomodachi ni tegami-o kaita -> friend to letter wrote*
  - **Lexical divergence:** words should be translated in context
    - Bass (the fish) vs bass (the instrument)

Common Approach

- **Encoder-decoder transformer**
  - Most common architecture for MT
- Often trained at sentence level
  - Get large set of parallel sentences (pairs in source and target languages)
  - Model maximizes probability of target sequence of tokens
  - Encoder produces context vector ***h***, decoder uses it to generate output

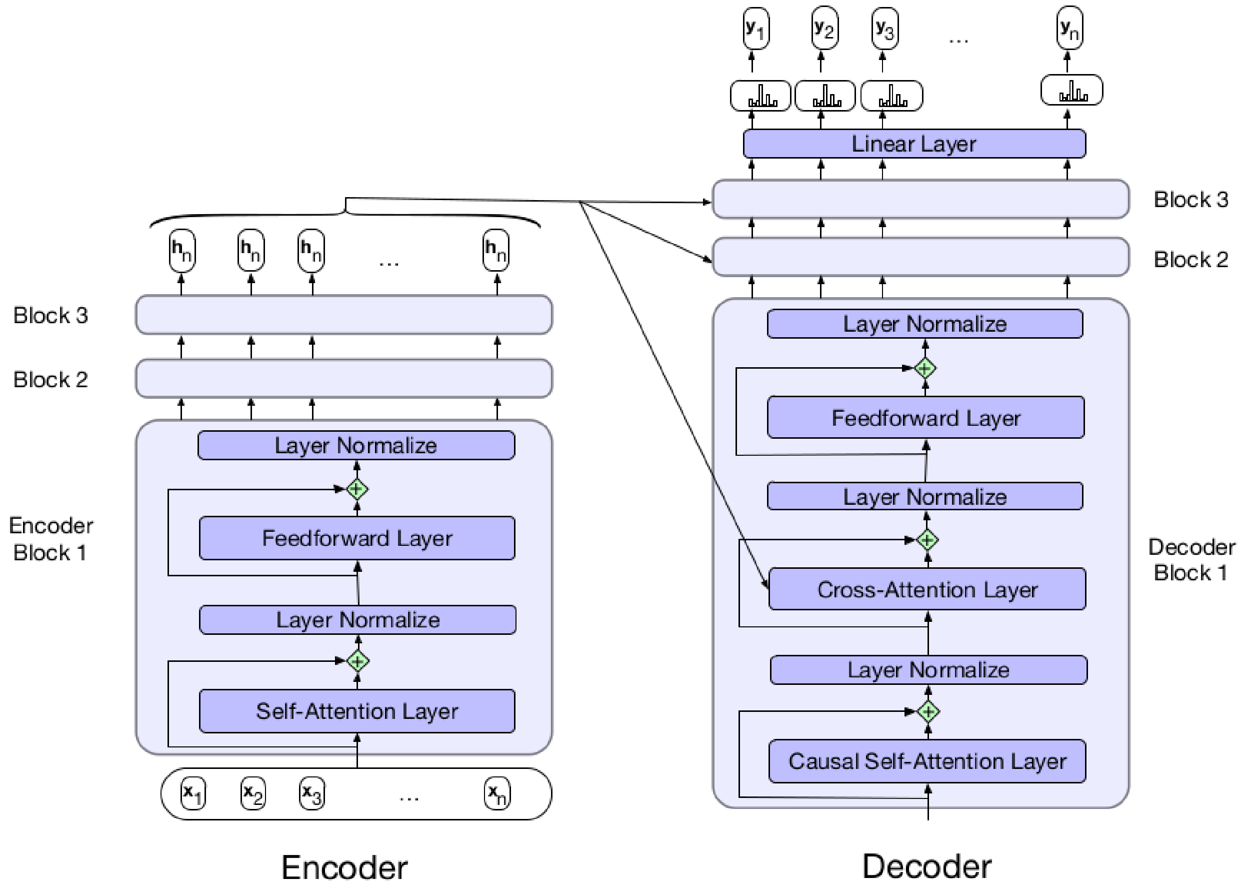
$$\mathbf{h} = encoder(x)$$

$$y_{i+1} = decoder(\mathbf{h}, y_1, y_2, \ldots, y_i)$$

- Training data requires **sentence alignment**
  - **Multilingual embedding space** used to check **sentence similarity**
- Common: use **shared vocabulary** between source and target language
  - **How?** Run tokenizer in corpus that contains both languages
  - Usually based on subwords, e.g. BytePair Encoding (BPE)
  - Recently, WordPiece and SentencePiece are used more often

Encoder-Decoder Architecture

- Important change: **modified cross-attention**
  - **Queries** come from previous layer (as usual), **keys/values** from encoder



MT Evaluation

- **Two common dimensions**
  - **Adequacy:** does translation capture exact meaning?
  - **Fluency:** is translation grammatically correct, clear, readable
- Two main approaches
  1. **Human**’s rate based on two dimensions (expensive, requires experience)
  2. Automatic methods
- **Automatic methods** often based on:
  - N-gram overlap with human translations (i.e. supervised)
  - Embedding similarity

What are Dialogue Systems?

- Programs that communicate with users in natural language
  - Often referred to as **conversational agents**
- **Two main categories**
  1. Task-oriented systems
  2. Chatbots
- **Task oriented** dialogue systems
  - **Goal:** use conversation to help complete tasks
  - E.g. give directions to control appliances, find restaurants, make calls, etc.
  - Common instance: **digital assistants**, e.g. Siri, Alexa, etc.
- **Chatbots**
  - **Goal:** designed for extended conversations
  - Needs to account for properties of human dialogue
  - E.g. we take turns, acknowledging understanding, making inferences, etc.

Chatbots

- Originally **rule-based**
  - E.g. Eliza would paraphrase what used said, turn it into therapy-like questions
- Others are **corpus-based**
  - Train on large corpus of billions of tokens of dialogue
  - Create answers via retrieval (fetch answer from DB) or generation
- **Hybrid** architectures
  - Combine rule-based with neural/corpus architectures
  - Arguably more reflective of real-world tools
  - E.g. Paranjape et al. (2020) built a system with **entity linking** to identify topic, **dialogue act classifier** to detect if user is asking question or making a statement, **user intent classifier** to detect if user wants to change topics
  - Response generation done with GPT-2 fine-tuned on dialogue dataset
- Let’s look at response generation in a bit more detail

Response Generation

- Two main approaches
  1. Retrieval-based
  2. Generation-based
- **Retrieval-based** answers
  - Train model on corpus *C* of dialogue (set of *"turns"*)
  - Treat user input as query *q*
  - Score each turn in *C* as potential answer to *q*, choose most likely one
- **Generation-based** answers
  - Modeled as **encoder-decoder** task
  - Encode user input with encoder, decoder generates response conditioned on user input + generated answer so far
  - Generation usually tuned to prevent most likely answers that may break the flow of conversation, e.g. "I don't know"
- Fine-tuning LLMs now common

Dialogue System Evaluation

- More straightforward if goal is perform specific task
  - E.g. was the restaurant booked? Was the event added to the calendar?
- Human-based approach: **user satisfaction rating**
  - Users interact with the system
  - Then fill-in multiple-choice questionnaire
- Since expensive, **heuristics often used**
  - They tend to correlate well with user satisfaction rating
- Heuristics often measure two things:
  - How well system allows user to achieve their goals with minimal cost?
  - **Task error rate**, e.g. how often was the event added to calendar after every interaction?

Summary

- Many more real-world applications:
  - Question Answering
  - Information Retrieval
  - RAG (Retrieval-enhanced LLMs)
  - Speech Recognition
  - Text-to-Speech
- Given an understanding of the fundamental methods:
  - Easy to understand architectures specific to given settings
  - Easy to reason about task/models/data

References

- Multiple Choice Normalization in LM Evaluation by Leo Gao
- Speech and Language Processing, Jurafsky and Martin, 2024
  - Chapters 13, 15
- The Evolving Landscape of LLM Evaluation by Sebastian Ruder
- References in corresponding slides