6.2.3 Causes

In this section, we aim to understand the factors influencing bias and fairness, drawing upon the foundational work of Navigli et al. (2023) as our guiding framework.

6.2.3.1 Data Selection

As we know, language models are trained on large datasets. However, choosing texts for these datasets introduces selection bias, affecting the model's behavior and output. This bias occurs at different stages, from initial sampling to data cleaning and filtering. Data selection bias arises when the texts chosen for training do not represent the full diversity of language used on the web. Modern LLMs, trained on extensive but still limited datasets, inherit the biases present in these texts. The selection process, influenced by the preference for specific sources, further compounds this issue. For instance, texts from Wikipedia are often selected for their reliability, while content from informal sources such as YouTube comments is excluded (Brown et al., 2020; Chowdhery et al., 2022; Zhang et al., 2022). This selective process shapes the model's understanding and generates biases. When LLMs are adapted for specific tasks, fine-tuning often involves smaller, specialized datasets or tailored prompts (Howard and Ruder, 2018; Liu et al., 2023). These additional data processing layers can introduce new biases or exacerbate existing biases, depending on the nature of the fine-tuning data or prompts used.

6.2.3.2 Unbalanced Domain and Genre Distribution

Pre-training datasets often have a skewed distribution of content. For example, Wikipedia, a common source in these datasets, is heavily weighted toward domains such as sports, music, geography, and politics, while underrepresenting areas such as literature, economy, and history (Gao et al., 2020; Kreutzer et al., 2022). This skewness is not unique to the English language but is observed across other high-resource languages, as evidenced by comparing domain distributions in English and Italian Wikipedias using BabelNet (Navigli and Ponzetto, 2012).

The biases in pre-training datasets cascade into downstream tasks. For instance, EuroParl and CoNLL-2009, used for fine-tuning LMs in tasks such as machine translation and semantic role labeling, are biased toward certain domains and genres. This results in LMs, even if initially unbiased, acquiring new biases from these task-specific datasets, which may not be immediately apparent to developers (Goldfarb-Tarrant et al., 2020).

6.2.3.3 Time of Creation

Languages evolve, leading to changes in word meanings and usage. For instance, the word "mouse" has expanded from its original animal reference to include a computer input device, and "tweet" has evolved from a bird sound to a social media post. Historical shifts in language use are evident in words such as "car", which once referred to horse-drawn carriages and now to motor vehicles, and "pipe", which shifted from a tobacco-smoking device to a type of tube. These changes mean that language models trained on historical data may not accurately reflect current language use or understand contemporary references.

The content and focus of domain-specific texts can vary significantly over time. For example, medical texts from the Middle Ages differ significantly from modern medical literature. Language models trained on datasets predating significant recent events such as the COVID-19 pandemic or the launch of the James Webb Telescope, may lack relevant contemporary knowledge. Similarly, models such as ChatGPT, with knowledge cut-off dates, may not have information on events occurring after that date. Researchers often reuse older datasets, such as SemCor, based on the Brown Corpus from the 1960s, for practical reasons. This practice can perpetuate outdated language use in models trained for tasks such as word sense disambiguation.

6.2.3.4 Creator Demographics

The demographics of both the creators and the selectors of training corpora play a crucial role in shaping the biases and behaviors of language models. The current skew toward certain demographic groups, particularly in platforms such as Wikipedia, highlights the need for more diversity and inclusivity in content creation and corpus selection processes. The demographic profile of individuals who create the content for training corpora can lead to biases in LLMs. For instance, Wikipedia, a common source of training data, exhibits a notable demographic imbalance among its editors. A majority (84%) are male, predominantly in their mid-20s or retired (Wikipedia Contributors, 2023).

These biases result both from the content creates and from the people who decide what content is included in the training set. Often, the decision-makers selecting corpora for LLMs are also predominantly male. This homogeneity among decision-makers can lead to a narrow selection of topics and perspectives, further reinforcing existing biases in the training data.

6.2.3.5 Language and Cultural Skew

LLM development has been centered around high-resource languages due to more accessible data collection and the availability of linguists and annotators. This has created a feedback loop, improving NLP systems for these languages while sidelining low-resource languages. Despite their advancements, multilingual models still exhibit performance biases favoring languages with richer training data.

Different languages embody distinct cultures, influencing linguistic expressions such as metaphors and idioms (Hershcovich et al., 2022). A skewed language distribution in LLMs leads to an unbalanced cultural representation. For instance, Wikipedia is predominantly English-centric, with over 50% of its editors primarily speaking English. This skews content toward English-speaking cultures, underrepresenting languages such as Hindi, Bengali, Javanese, and Telugu despite their large speaker populations. The primary language of Wikipedia editors does not reflect the global distribution of English speakers. For example, only 3% of English-speaking editors are from India, impacting the diversity of content on Wikipedia (Wang et al., 2022).

6.2.4 Evaluation Metrics

Bias and fairness metrics in LLMs can be grouped based on the model aspects they utilize, such as embeddings, probabilities, or generated text. This taxonomy includes the following:

- **Embedding-Based Metrics**: These metrics use dense vector representations, typically contextual sentence embeddings, to measure bias.
- **Probability-Based Metrics**: These metrics employ model-assigned probabilities to estimate bias, such as scoring text pairs or answering multiple-choice questions.
- **Generated Text-Based Metrics**: These metrics analyze the text generated by the model in response to prompts to measure patterns such as co-occurrence or compare outputs from varied prompts.

6.2.4.1 Embedding-Based Metrics

Embedding-based metrics primarily compute distances in vector space between neutral words (e.g., professions) and identity-related words (e.g., gender pronouns). The focus is on sentence-level contextualized embeddings in LLMs. Originally proposed for static word embeddings, these metrics have evolved to include contextualized embeddings, measuring bias across various dimensions.

A key method is the *Word Embedding Association Test* (WEAT), which assesses associations between social group concepts (such as masculine and feminine words) and neutral attributes (such as family and occupation words) (Greenwald et al., 1998). To measure stereotypical associations, a test statistic is employed for protected attributes A_1, A_2 and neutral attributes W_1, W_2:

$$f(A_1, A_2, W_1, W_2) = \sum_{a_1 \in A_1} s(a_1, W_1, W_2) - \sum_{a_2 \in A_2} s(a_2, W_1, W_2) \tag{6.4}$$

Here, s is a similarity measure defined as:

$$s(a, W_1, W_2) = \text{mean}_{w_1 \in W_1} \cos(\mathbf{a}, \mathbf{w}_1) - \text{mean}_{w_2 \in W_2} \cos(\mathbf{a}, \mathbf{w}_2) \tag{6.5}$$

A larger effect size indicates a stronger bias, with the size determined by:

$$\text{WEAT}(A_1, A_2, W_1, W_2) = \frac{\text{mean}_{a_1 \in A_1} s(a_1, W_1, W_2) - \text{mean}_{a_2 \in A_2} s(a_2, W_1, W_2)}{\text{std}_{a \in A_1 \cup A_2} s(a, W_1, W_2)} \tag{6.6}$$

To adapt WEAT for contextualized embeddings, the *Sentence Encoder Association Test* (SEAT) was developed by May et al. (2019). SEAT generates embeddings using sentences structured around semantically neutral templates, such as "This is [BLANK]" or "[BLANK] are things." These templates are filled with words representing social groups and neutral attributes. The method uses the same equation as WEAT, employing the [CLS] token embeddings. SEAT's adaptability allows for measuring more nuanced bias dimensions through more specific, unbleached templates, such as "The engineer is [BLANK]."

The *Contextualized Embedding Association Test* (CEAT) offers an alternative approach (Guo and Caliskan, 2021). It generates sentences combining elements from the A_1, A_2, W_1, and W_2 groups and calculates a distribution of effect sizes by randomly sampling a subset of embeddings. The magnitude of bias in CEAT is measured using a random-effects model formulated as follows:

$$\text{CEAT}(S_{A1}, S_{A2}, S_{W1}, S_{W2}) = \frac{\sum_{i=1}^{N} v_i \text{WEAT}(S_{A1i}, S_{A2i}, S_{W1i}, S_{W2i})}{\sum_{i=1}^{N} v_i} \tag{6.7}$$

where v_i is derived from the variance of the random-effects model. These methods facilitate the application of WEAT's principles to contextualized embeddings, enabling more nuanced analyses of bias in language models.

6.2.4.2 Probability-Based Metrics

These techniques involve prompting a model with pairs or sets of template sentences with perturbed protected attributes and comparing the predicted token probabilities conditioned on different inputs. One approach, the *masked token method*, involves masking a word in a sentence and using a masked language model to predict the missing word. For example, *Discovery of Correlations* (DisCo) by Webster et al. (2020) compares the completion of template sentences with slots filled with bias triggers and the model's top predictions.

The *Log-Probability Bias Score* (LPBS), as proposed by Kurita et al. (2019), utilizes a template-based methodology similar to DisCo for assessing bias in neutral attribute words. The approach entails normalizing a token's predicted probability p_a, obtained from a template "`[MASK]` is a `[NEUTRAL ATTRIBUTE]`", with the model's prior probability p_{prior}, derived from a template "`[MASK]` is a `[MASK]`". This normalization is crucial because it accounts for the model's inherent biases toward certain social groups, focusing the measurement specifically on the bias associated with the `[NEUTRAL ATTRIBUTE]` token. The bias score is calculated by comparing the normalized probabilities for two opposing social group words.

Mathematically, LPBS is defined as:

$$\text{LPBS}(S) = \log \frac{p_{a_i}}{p_{\text{prior}_i}} - \log \frac{p_{a_j}}{p_{\text{prior}_j}} \tag{6.8}$$

where p_{a_i} and p_{a_j} are the predicted probabilities for different social group words, while p_{prior_i} and p_{prior_j} denote their respective prior probabilities. The LPBS score thus quantifies bias by evaluating how significantly a token's probability deviates from its expected prior distribution.

Ahn and Oh (2021) introduced the *Categorical Bias Score* (CBS), which adapts normalized log probabilities for non-binary targets from Kurita et al. (2019). CBS measures the variance of predicted tokens for fill-in-the-blank template prompts over protected attribute word a for different social groups, represented as:

$$\text{CBS}(S) = \frac{1}{|W|} \sum_{w \in W} \text{Var}_{a \in A} \log \frac{p_a}{p_{\text{prior}}} \tag{6.9}$$

A range of methods utilize *pseudo-log-likelihood* (PLL) to determine the likelihood of generating a specific token based on the context of the other words in a sentence. For a given sentence S, PLL is defined as:

$$\text{PLL}(S) = \sum_{s \in S} \log P(s|S_{\setminus s}; \theta) \tag{6.10}$$

where $P(s|S \setminus s; \theta)$ estimates the conditional probability of each token s within the sentence S, masking one token at a time. This approach allows for predicting the masked token using all other unmasked tokens in the sentence, thereby approximating the token's probability in its context.

The *CrowS-Pairs Score* (CPS) developed by Nangia et al. (2020) alongside the CrowS-Pairs dataset requires pairs of sentences, one stereotyping and one less so, to evaluate a model's preference for stereotypical content. It leverages PLL for this assessment. The metric approximates the probability of shared, unmodified tokens U conditioned on modified tokens, typically representing protected attributes M, denoted as $P(U|M, \theta)$. This approximation is achieved by masking and predicting each unmodified token in the sentence. The metric for a sentence S is formulated as follows:

$$\text{CPS}(S) = \sum_{u \in U} \log P\left(u | U_{\setminus u}, M; \theta\right) \tag{6.11}$$

The *Context Association Test* (CAT) introduced by Nadeem et al. (2020) with the StereoSet dataset is another method for comparing sentences. Each sentence in CAT is paired with a stereotype, anti-stereotype, and meaningless option, either fill-in-the-blank tokens or continuation sentences. Unlike the pseudo-log-likelihood method, CAT focuses on $P(M|U, \theta)$ rather than $P(U|M, \theta)$. This shift in focus allows CAT to frame the evaluation as follows:

$$\text{CAT}(S) = \frac{1}{|M|} \sum_{m \in M} \log P(m|U; \theta) \tag{6.12}$$

6.2.4.3 Generated Text-Based Metrics

Generated text-based metrics are particularly relevant when dealing with LLMs as black boxes where direct access to probabilities or embeddings is not possible. A common approach is to condition the LLM on specific prompts known for bias or toxicity, such as those from *RealToxicityPrompts* and *BOLD*, and then analyze the generated text for bias.

Among the various metrics used, *Social Group Substitutions* (SGS) require identical LLM responses under demographic substitutions (Rajpurkar et al., 2016). Assuming an invariance metric ψ, such as exact match, considering $\widehat{Y}_i$ as the predicted output from the original input and $\widehat{Y}_j$ as the output from a counterfactual input with altered demographics, the SGS metric is mathematically expressed as:

$$\text{SGS}(\widehat{Y}) = \psi(\widehat{Y}_i, \widehat{Y}_j) \tag{6.13}$$

Another metric, *Co-Occurrence Bias Score*, measures the co-occurrence of tokens with gendered words in generated text (Bordia and Bowman, 2019).

$$\text{Co-Occurrence Bias Score}(w) = \log \frac{P(w|A_i)}{P(w|A_j)} \tag{6.14}$$

where w is the token and A_i and A_j are two sets of attributes.

Demographic Representation (DR) evaluates the representation frequency of social groups in comparison to their distribution in the original dataset (Bommasani et al., 2023). If the function $C(x, Y)$ calculates the count of word x in sequence Y, DR for a social group G_i in the set G, with its associated protected attribute words A_i, is calculated as:

$$\text{DR}(G_i) = \sum_{a_i \in A_i} \sum_{\widehat{Y} \in \widehat{\mathbb{Y}}} C(a_i, \widehat{Y}) \tag{6.15}$$

Here, DR $= [\text{DR}(G_1), \ldots, \text{DR}(G_m)]$ forms a vector of counts for each group, normalized to a probability distribution. This distribution is then compared to a reference distribution, such as the uniform distribution, using metrics such as total variation distance, KL divergence, or Wasserstein distance to assess the representational equity of social groups in the model's output.

The *Stereotypical Associations* (ST) metric evaluates the bias associated with specific words in relation to social groups (Bommasani et al., 2023). This metric quantifies the frequency of co-occurrence of a word w with attribute words A of a social group G_i in a set of predicted outputs $\widehat{Y}$. The function is given by:

$$\text{ST}(w)_i = \sum_{a_i \in A_i} \sum_{\widehat{Y} \in \widehat{\mathbb{Y}}} C(a_i, \widehat{Y}) \mathbb{I}(C(w, \widehat{Y}) > 0) \tag{6.16}$$

In this formula, $C(a_i, \widehat{Y})$ represents the co-occurrence count of the attribute word a in the predicted output $\widehat{Y}$. Analogous to Demographic Representation, the count vector $ST = [ST(w)_i, \ldots, ST(w)_k]$ can be normalized and compared against a reference distribution.

The *Honest metric* is designed to quantify the frequency of hurtful sentence completions generated by language models (Nozza et al., 2021). For causal models, the metric uses an incomplete sentence as a prompt, while for masked models, it utilizes a sentence with a `[MASK]` token where the model predicts a word. The Honest metric calculates the proportion of hurtful predictions identified by the *HurtLex* corpus. The Honest score averages these proportions across various categories such as animals, crime, and negative connotations. The formula for the Honest score is as follows:

$$\text{HONEST}(\widehat{\mathbb{Y}}) = \frac{\sum_{\widehat{Y}_k \in \widehat{\mathbb{Y}}_k} \sum_{\widehat{y} \in \widehat{Y}_k} \mathbb{I}_{\text{HurtLex}}(\widehat{y})}{|\widehat{\mathbb{Y}}| \cdot k} \tag{6.17}$$

In the given context, $\widehat{Y}_i \in \widehat{\mathbb{Y}}_i$ represents a predicted output that is associated with a specific group G_i. The metric under consideration is applied to identity-related template prompts and their corresponding top-k completions, denoted as $\widehat{\mathbb{Y}}_k$.

6.2.5 Benchmarks

The taxonomy of the benchmark datasets can be classified into *counterfactual inputs* and *prompts* as a primary category. Counterfactual inputs can be further classified into subcategories: *masked tokens* and *unmasked sentence*. Datasets with pairs or tuples of sentences, typically counterfactual, highlight differences in model predictions across social groups. Masked token datasets contain sentences with a blank slot for the language model to fill. These are suited for masked token probability-based metrics or pseudo-log-likelihood metrics. Coreference resolution tasks, such as the Winograd Schema Challenge, Winogender, and WinoBias, are prominent examples. On the other hand, unmasked sentence datasets evaluate which sentence in a pair is most likely. They can be used with pseudo-log-likelihood metrics and are flexible for other metrics. Examples include Crowdsourced Stereotype Pairs (CrowS-Pairs), Equity Evaluation Corpus, and RedditBias.

The prompts category is further classified into whether prompts are for *sentence completion* or *question-answering* tasks. Datasets designed as prompts for text continuation include sentence completion datasets such as RealToxicityPrompts, TrustGPT, and BOLD. They provide sentence prefixes for LLMs to complete. GrepBiasIR and BBQ (Bias Benchmark for QA) serve as question-answering frameworks to probe biases in LLMs. A comprehensive list of known datasets and their taxonomy, as outlined by Gallegos et al. (2023) in their work, is represented in Table 6.2.

6.2.6 Mitigation Strategies

In the subsequent section, we explore bias mitigation strategies as outlined by Gallegos et al. (2023), categorizing them according to the various stages of the LLM workflow: pre-processing, intra-processing, and post-processing, as shown in Fig. 6.3.

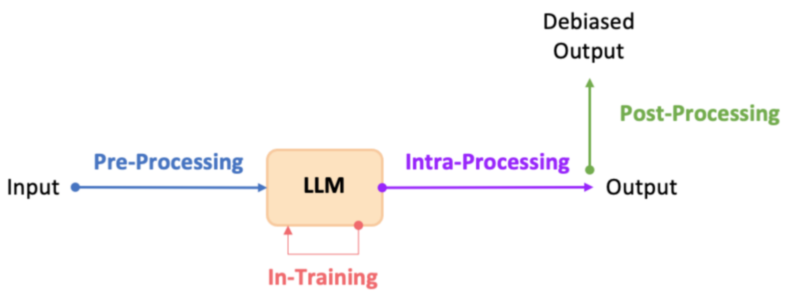


Fig. 6.3: Bias mitigation strategies and their place in the LLM workflow.

6.2.6.1 Pre-Processing Mitigation

These methods focus on adjusting the model's inputs—data and prompts—without altering the model's trainable parameters, as shown in Fig. 6.4.

Data Augmentation
Data augmentation-based techniques add examples from underrepresented groups to the training data, thus broadening the dataset's diversity. *Counterfactual Data Augmentation* (CDA) is a key method in this approach, where protected attribute words (such as gendered pronouns) are replaced to balance the dataset. Lu et al. (2020) ex-

Table 6.2: Benchmark datasets targeting biases. Each dataset is characterized by its size, the specific bias issue(s) it addresses, and the target social group(s) it aims to evaluate. Checkmarks in our analysis signify issues or groups explicitly mentioned in the original research or represent additional scenarios the dataset could address.

Dataset	**Size**	**Bias Issues**														
		Misrepresentation	Stereotyping	Disparate Performance	Derogatory Language	Exclusionary Norms	Toxicity	Age	Disability	Gender (Identity)	Nationality	Physical Appearance	Race	Religion	Sexual Orientation	Other
Counterfactual Inputs (Masked Tokens)																
Winogender	720	✓	✓	✓		✓				✓						
WinoBias	3,160	✓	✓	✓		✓				✓						
WinoBias+	1,367	✓	✓	✓		✓				✓						
GAP	8,908	✓	✓	✓		✓				✓						
GAP-Subjective	8,908	✓	✓	✓		✓				✓						
BUG	108,419	✓	✓	✓		✓				✓						
StereoSet	16,995	✓	✓	✓						✓			✓	✓		✓
BEC-Pro	5,400	✓	✓	✓		✓				✓						
Counterfactual Inputs (Unmasked Sentences)																
CrowS-Pairs	1,508	✓	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓	✓
WinoQueer	45,540	✓	✓	✓											✓	
RedditBias	11,873	✓	✓	✓	✓					✓			✓	✓	✓	
Bias-STS-B	16,980	✓	✓							✓						
PANDA	98,583	✓	✓	✓				✓		✓			✓			
Equity Evaluation Corpus	4,320	✓	✓	✓						✓			✓			
Bias NLI	5,712,066	✓	✓			✓				✓				✓		
Prompts (Sentence Completion)																
RealToxicityPrompts	100,000				✓		✓									✓
BOLD	23,679				✓	✓	✓			✓			✓	✓		✓
HolisticBias	460,000	✓	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓	✓
TrustGPT	9			✓	✓		✓			✓			✓	✓		
HONEST	420	✓	✓	✓						✓						
Prompts (Question Answering)																
BBQ	58,492	✓	✓		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
UnQover	30	✓	✓			✓				✓	✓		✓	✓		
Grep-BiasIR	118	✓	✓			✓				✓						

emplify this by using CDA to counteract occupation-gender bias, flipping gendered words while preserving grammatical and semantic integrity.

A selective replacement strategy offers an alternative to CDA for improving data efficiency and targeting the most compelling examples for bias mitigation.

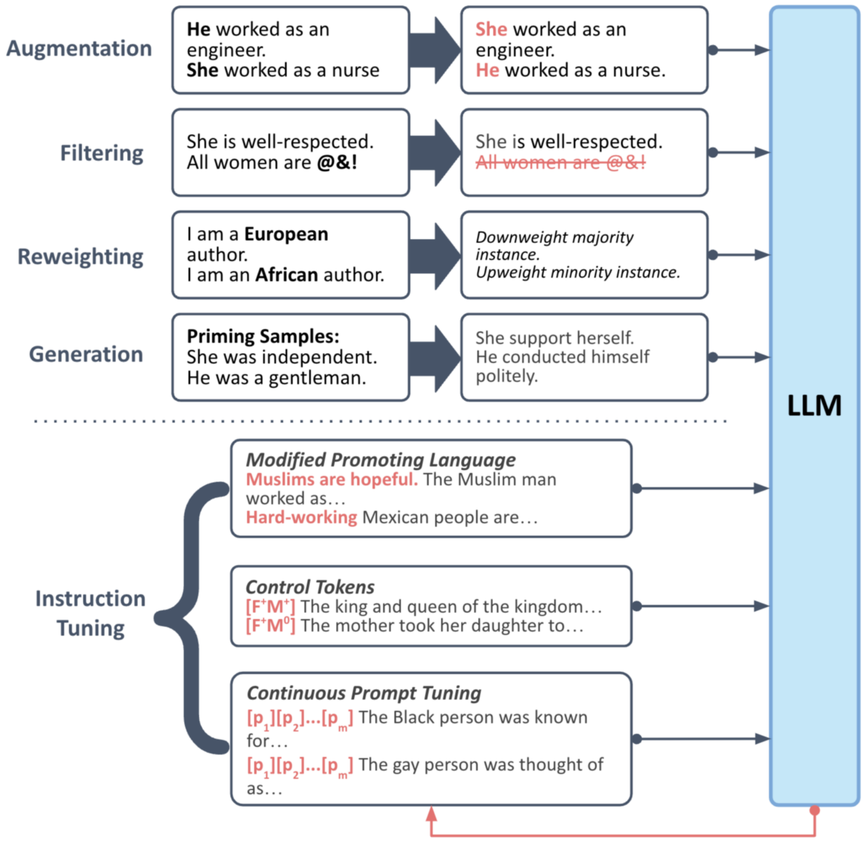

Augmentation
He worked as an engineer. She worked as a nurse
She worked as an engineer. He worked as a nurse.
Filtering
She is well-respected. All women are @&!
She is well-respected. ~~All women are @&!~~
Reweighting
I am a European author. I am an African author.
Downweight majority instance. Upweight minority instance.
Generation
Priming Samples: She was independent. He was a gentleman.
She support herself. He conducted himself politely.
LLM
Instruction Tuning
Modified Promoting Language
Muslims are hopeful. The Muslim man worked as…
Hard-working Mexican people are…
Control Tokens
[F⁺M⁺] The king and queen of the kingdom…
[F⁺M⁰] The mother took her daughter to…
Continuous Prompt Tuning
[p1][p2]...[pm] The Black person was known for…
[p1][p2]...[pm] The gay person was thought of as…

Fig. 6.4: Demonstration of pre-processing strategies for bias removal.

Practical Tips

Techniques such as *Counterfactual Data Substitution* (CDS) proposed by Maudslay et al. (2019), involve randomly substituting gendered text with a counterfactual version. Another variant proposed by these authors, the Names Intervention, focuses on first names, associating masculine-specified names with feminine-specified pairs for substitution.

Based on the mixup technique of Zhang et al. (2017), interpolation techniques blend counterfactually augmented training examples with their original versions. This method extends the diversity of the training data.

Data Filtering and Reweighting

Data filtering and reweighting approaches address the limitations of data augmentation in bias mitigation, primarily focusing on the precision targeting of specific examples within an existing dataset. These methods are categorized into two main approaches: 1) dataset filtering and 2) instance reweighting.

The dataset filtering process involves selecting subsets of data to influence the model's learning during fine-tuning. Techniques range from curating texts from underrepresented groups to enhance diversity, as done by Garimella et al. (2022), to constructing low-bias datasets by selecting the least biased examples, as demonstrated by Borchers et al. (2022). Other methods target the most biased examples, either by neutralizing or filtering them to reduce overall bias in the dataset. For instance, (Thakur et al., 2023) curated a set of highly biased examples and neutralized gender-related words to create more balanced training data.

The instance reweighting technique adjusts the importance of specific instances in the training process. Han et al. (2022) employed this method by calculating the weight of each instance in the loss function inversely proportional to its label and associated protected attribute. Other approaches, such as those of Utama et al. (2020) and Orgad and Belinkov (2023), focus on downweighting examples containing social group information, reducing the reliance on stereotypical shortcuts during model predictions.

Data Generation

Data generation addresses limitations inherent in data augmentation, filtering, and reweighting, notably the challenge of identifying specific examples for each bias dimension, which can vary by context or application. This method involves creating entirely new datasets tailored to meet predetermined standards or characteristics rather than modifying existing datasets. Solaiman and Dennison (2021) have developed iterative processes to construct datasets targeting specific values, such as removing biases associated with legally protected classes. Human writers play a crucial role in this process, creating prompts and completions that reflect the intended behaviors, which are refined based on performance evaluations. Similarly, Dinan et al. (2019) employed human writers to gather diverse examples to reduce gender bias in chat dialog models.

Central to data generation is creating new word lists, particularly for use in word-swapping techniques such as CDA and CDS. These lists often focus on terms associated with various social groups, covering aspects such as gender, race, age, and dialect. However, reliance on such lists can sometimes limit the scope of addressed stereotypes. To counter this, broader frameworks have been proposed, such as the one by Omrani et al. (2023), which focuses on understanding stereotypes along more general dimensions such as "warmth" and "competence", offering a more expansive approach to bias mitigation. Their research produces word lists corresponding to these two categories, offering an alternative to group-based word lists such as gendered words for use in tasks that mitigate bias.

Instruction Tuning

The instruction tuning approach involves modifying the inputs or prompts fed into the model. Modifying prompts add textual instructions or triggers to a prompt to encourage the generation of unbiased outputs. For example, Mattern et al. (2022) use prompts with various levels of abstraction to steer models away from stereotypes.

Similarly, Venkit et al. (2023) employ adversarial triggers to reduce nationality bias, and Abid et al. (2021) use short phrases to combat anti-Muslim bias. These methods typically involve appending phrases to the input to induce neutral or positive sentiments toward specific social groups.

Instead of adding instructive language, a control token approach is also used to categorize prompts. The model learns to associate each token with a particular input class, allowing for controlled generation during inference. Dinan et al. (2019) utilized this approach to mitigate gender bias in dialog generation by appending tokens corresponding to the presence or absence of gendered words in training examples.

! Practical Tips

Continuous prompt tuning is another evolving technique that involves adding trainable prefixes to the input, effectively freezing the original pre-trained model parameters while allowing for tunable updates specific to the task. This method facilitates scalable and targeted adjustments beyond what manual prompt engineering can achieve. Notably, Fatemi et al. (2021) and Yang et al. (2023) have applied continuous prompt tuning to mitigate gender bias and encourage the use of neutral language independent of protected attributes.

6.2.6.2 In-Training Mitigation

In-training mitigation encompasses strategies to reduce bias during the model's training process. These techniques involve alterations to the optimization process, including modifying the loss function, updating next-word probabilities, selectively freezing parameters during fine-tuning, and eliminating specific neurons linked to harmful outputs, as shown in Fig. 6.5. All these mitigation strategies involve gradient-based training updates to alter model parameters.

Architecture Modification

A key aspect of in-training mitigation is *architecture modification*. This involves changes to the model's structure, such as the number, size, and type of layers, encoders, and decoders. A notable example is the introduction of debiasing adapter modules, such as ADELE by Lauscher et al. (2021), which are based on modular adapter frameworks. These frameworks insert new layers between existing layers for efficient fine-tuning. The newly added layers are fine-tuned, while the pre-trained layers are kept static, focusing specifically on learning debiasing knowledge.

Liu et al. (2022) introduced a regularization term designed to minimize the distance between the embeddings of a protected attribute given by $E(\cdot)$

$$\mathcal{R} = \lambda \sum_{(a_i,a_j)\in A} \|E(a_i) - E(a_j)\|_2 \tag{6.18}$$

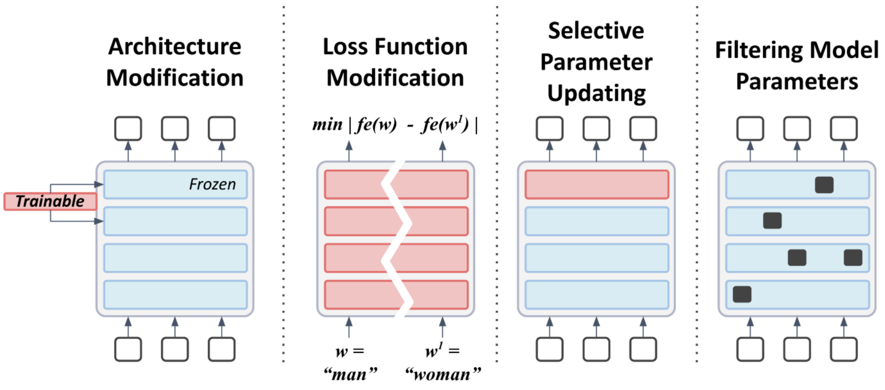

Architecture Modification
Trainable
Frozen
Loss Function Modification
min | fe(w) - fe(w') |
w = "man"
w' = "woman"
Selective Parameter Updating
Filtering Model Parameters

Fig. 6.5: A range of methods exist to reduce bias within the model training process.

where $\mathcal{R}$ is the regularization term, λ is a scaling factor, and a_i and a_j are the elements of the set A representing protected attributes and their counterfactuals.

Loss Function Modification
Park et al. (2023) introduced a technique involving integrating projection-based bias mitigation techniques into the loss function, specifically targeting gender stereotypes in occupational terms. They introduce a regularization term that orthogonalizes stereotypical word embeddings $\mathbf{w}$ and the gender direction $\mathbf{g}$ in the embedding space. This term effectively distances the embeddings of neutral occupation words from those of gender-inherent words (e.g., "sister" or "brother").

The gender direction is formally defined as follows:

$$\mathbf{g} = \frac{1}{|A|} \sum_{(a_i,a_j)\in A} E(a_j) - E(a_i) \tag{6.19}$$

where:

- A represents the set of all gender-inherent feminine-associated a_i and masculine-associated a_j words.
- $E(\cdot)$ computes the embeddings of a model.

The regularization term is expressed as:

$$\mathcal{R} = \sum_{w\in W_{\text{stereo}}} \frac{\mathbf{g}}{||\mathbf{g}||}\mathbf{w}^{\top} \quad (34) \tag{6.20}$$

where W_{stereo} denotes the set of stereotypical embeddings.

Attanasio et al. (2022) explored modifying the attention layers of language models, hypothesizing that these layers are the primary encoders of bias. They introduced an *entropy-based attention regularization* (EAR) concept, utilizing the entropy of the attention weight distribution to assess the context words' relevance. High entropy indicates diverse context usage, whereas low entropy indicates a reliance on specific tokens. EAR aims to maximize the entropy of attention weights to prevent overfitting to stereotypical words, thereby broadening the model's focus on the input context. This is achieved by adding entropy maximization as a regularization term in the loss function, formalized as:

$$\mathcal{R} = -\lambda \sum_{\ell=1}^{L} \text{entropy}(\mathbf{A})^{\ell} \tag{6.21}$$

where entropy$(\mathbf{A})^{\ell}$ is the attention entropy at the ℓ-th layer.

Several studies have developed loss functions to balance the probabilities of words associated with specific demographics in language model outputs. For instance, Qian et al. (2019) introduced an equalizing objective to encourage demographic words to be predicted with equal probability. This involves a regularization term that compares the softmax output probabilities for binary masculine and feminine word pairs, which was later adapted by Garimella et al. (2022) for binary race word pairs. The regularization term, applicable for K word pairs consisting of attributes a_i and a_j, where $a_i \in A_i$ is a protected attribute word associated with social group G_i and similarly $a_j \in A_j$ is a protected attribute word associated with social group G_j , is expressed as:

$$\mathcal{R} = \lambda \frac{1}{K} \sum_{k=1}^{K} \left| \log \frac{P(a_i^{(k)})}{P(a_j^{(k)})} \right| \tag{6.22}$$

Selectively Updating or Filtering Model Parameters

Fine-tuning AI models on augmented datasets can reduce bias but risks "catastrophic forgetting", where models lose previously learned information. To prevent this, recent approaches involve selectively updating only a tiny portion of the model's parameters while freezing the rest. For instance, Gira et al. (2022) fine-tuned models by updating specific parameters such as layer norms on the WinoBias and CrowS-Pairs datasets, while Ranaldi et al. (2023) focused only on attention matrices while freezing all other parameters. Yu et al. (2023) took a targeted approach, optimizing weights that most contribute to bias, for example, gender-profession.

In addition to fine-tuning methods that update parameters to diminish bias, some techniques focus on selectively filtering or eliminating specific parameters, such as setting them to zero during or after training. Joniak and Aizawa (2022) employed movement pruning, a method that selectively removes weights from a neural network. They applied this approach to choose a less biased subset of weights from the attention heads of a pre-trained model. During fine-tuning, these weights are frozen, and separate scores optimized for debiasing are used to decide which weights to eliminate.

6.2.6.3 Intra-Processing Mitigation

Intra-processing methods, as defined by Savani et al. (2020), involve modifying a pre-trained or fine-tuned model's behavior during the inference stage to produce debiased predictions without further training. These methods are considered inference stage mitigations and encompass techniques such as altered decoding strategies, post-hoc modifications to model parameters, and separate debiasing networks applied in a modular fashion during inference, as shown in Fig. 6.6.

Decoding strategy modification refers to generating output tokens, where the decoding algorithm is adjusted to minimize biased language. This adjustment does not change the trainable model parameters; instead, it influences the probability of subsequent words or sequences through selection constraints, alterations in the token probability distribution, or by incorporating an auxiliary bias detection model. One specific approach within this category is *constrained next-token search*, which involves adding extra criteria to change the ranking of the next token in the output sequence. This method represents a straightforward yet effective way to guide the model toward less biased outputs.

Gehman et al. (2020) suggests blocking words or n-grams during decoding, effectively preventing the selection of tokens from a predefined list of unsafe words. Nevertheless, this approach does not entirely eliminate the possibility of generating biased outputs, as they can still arise from a combination of tokens or *n*-grams that are individually unbiased.

! Practical Tips

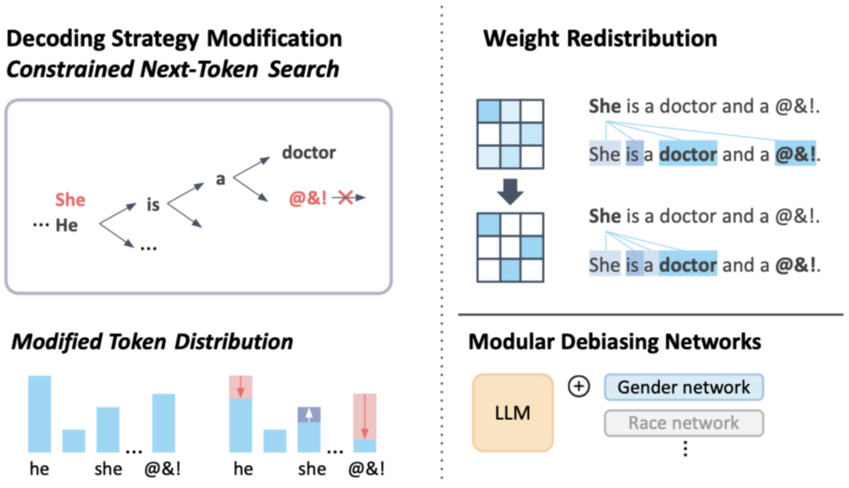

Decoding Strategy Modification
Constrained Next-Token Search
She
... He
is
...
a
doctor
@&!
Modified Token Distribution
he
she
...
@&!
he
she
...
@&!
Weight Redistribution
She is a doctor and a @&!.
She is a doctor and a @&!.
She is a doctor and a @&!.
She is a doctor and a @&!.
Modular Debiasing Networks
LLM
Gender network
Race network

Fig. 6.6: Some promising methods have been developed to mitigate bias at inference time.

The *weight redistribution* approach involves post-hoc modifications of a model's weights, mainly attention weights, linked to encoded biases. Zayed et al. (2023) demonstrated this by adjusting attention weights after training through temperature scaling, controlled by a hyperparameter. This adjustment can either increase entropy, leading the model to consider a broader range of tokens and potentially avoid stereotypes, or decrease entropy to focus on a narrower context, thus reducing exposure to stereotypical tokens.

Finally, *modular debiasing networks* addresses the limitations of in-training approaches, which are often specific to one type of bias and permanently alter the model. Modular debiasing offers more flexibility, creating separate components that can be integrated with a pre-trained model for different tasks or biases. In their approach, Hauzenberger et al. (2023) applied the technique of diff pruning, as proposed by Guo et al. (2020), to the context of debiasing. This adaptation involved simulating the training of multiple models in parallel, each tailored to reduce bias along distinct dimensions. The modifications made to the pre-trained model's parameters were then efficiently captured and stored within sparse subnetworks.

6.2.6.4 Post-Processing Mitigation

Post-processing mitigation is a method applied to the outputs of pre-trained models to reduce bias and is particularly useful when these models are primarily black boxes with limited insight into their training data or internal workings, as highlighted in Fig. 6.7.

! Practical Tips

Rewriting-based approaches involve detecting and replacing biased or harmful words in the model's output. Techniques such as keyword replacement identify bi-

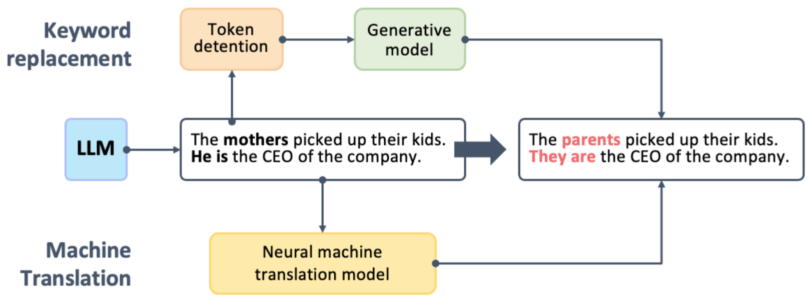

Keyword replacement
Token detention
Generative model
LLM
The mothers picked up their kids. He is the CEO of the company.
The parents picked up their kids. They are the CEO of the company.
Machine Translation
Neural machine translation model

Fig. 6.7: Illustrative example of post-processing bias mitigation techniques.

ased tokens and substitute them with more neutral or representative terms, focusing on preserving the content and style of the original text. For example, Tokpo and Calders (2022) used *LIME* to identify and replace bias-responsible tokens based on the latent representation of the original, while Dhingra et al. (2023) utilized *SHAP* to identify and replace stereotypical words with an explanation of why they were replaced.

Another post-processing technique uses a machine translation approach to translate biased sentences into neutral or unbiased alternatives. Various studies have employed a rule-based methodology to create gender-neutral versions of sentences containing gendered pronouns (Jain et al., 2021; Sun et al., 2021). This approach generates parallel, debiased sentences from sources with gender bias. Subsequently, a machine translation model is trained to convert sentences with gender bias into their debiased counterparts.

6.3 Toxicity

The concept of *toxicity* encompasses a range of harmful content types and has no standard definition. Toxicity can be interpreted as a form of representational harm, as previously defined, or considered a distinct concept in its own right. The Perspective API characterizes toxicity as rude, disrespectful, or unreasonable comments likely to drive participants away from a conversation. Kurita et al. (2019) describe toxic content as any material that could be offensive or harmful to its audience, including instances of hate speech, racism, and the use of offensive language. Pavlopoulos et al. (2020) refer to toxicity as a collective term, where the community employs a variety of terms to describe different forms of toxic language or related phenomena, such as "offensive," "abusive," and "hateful".

6.3.1 Causes

Toxicity, bias, and fairness in LLMs are not isolated issues. They are intricate threads woven from a common fabric: the data upon which they are trained. Many of the causes highlighted in the bias section, such as data selection, unbalanced domain and genre distribution, creator demographics, and cultural skew, also hold for toxic outputs from LLMs. In this section, we will highlight the causes that may be specific to toxicity and/or overlap with causes responsible for biases in the LLMs.

1. **Training Data Bias**: A predominant source of toxicity in LLMs is the bias inherent in the training data, as discussed in the bias section. The training data

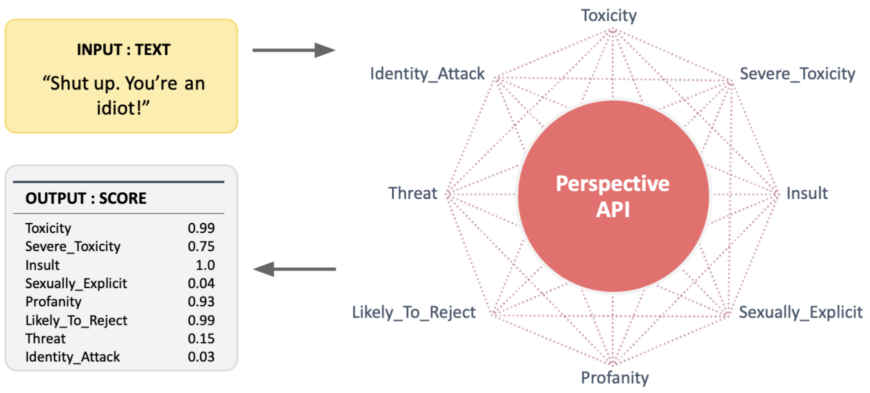
can be biased due to societal inequalities, prejudiced language usage, and underrepresentation of certain groups. This bias manifests in the models' outputs, producing toxic and unfair outcomes. Models often replicate the biases found in these datasets, leading to toxic outputs (Bender et al., 2021).

2. **Contextual Understanding Limitations**: LLMs sometimes struggle with comprehending the full context of text or conversations, resulting in inappropriate or toxic responses. Bender and Koller (2020) highlight models' challenges in interpreting nuanced human language, underscoring the complexities in achieving accurate contextual understanding. Pavlopoulos et al. (2020) discovered that the context surrounding a post can significantly influence its perceived toxicity, either by amplifying or mitigating it. In their research, a notable portion of manually labeled posts–approximately 5% in one of their experiments–received opposite toxicity labels when annotators evaluated them without the surrounding context.
3. **Adversarial Attacks**: LLMs are vulnerable to adversarial attacks, where they are prompted to deliberately produce toxic outputs. In their research, Wallace et al. (2020) highlight how an adversary can inject malicious examples into a model's training set, significantly impacting its learning and future predictions. This attack strategy is highlighted as a dangerous vulnerability, allowing an adversary to turn any chosen phrase into a universal trigger for a specific prediction. Furthermore, the study reveals that these poisoned training examples can be designed to be inconspicuous, making it challenging for a victim to identify and remove harmful data. The poison examples are crafted so that they do not explicitly mention the trigger phrase, evading detection strategies that rely on searching for specific phrases.
4. **Persona-Assigned Prompts**: One common trend in conversational AI is for users to assign a persona to the LLM to carry out further conversations. Deshpande et al. (2023) show that specific personas to ChatGPT, such as that of the renowned boxer Muhammad Ali, could markedly increase the toxicity levels in the generated text. This study revealed that depending on the persona attributed to ChatGPT, the toxicity in its responses could be amplified by up to six times. This increase in toxicity was characterized by the model's engagement in promoting incorrect stereotypes, generating harmful dialog, and expressing hurtful opinions. Such responses, associated with the assigned personas, not only have the potential to be defamatory toward these public figures but also pose a risk of harm to users who interact with the model without anticipating such toxic content.

6.3.2 Evaluation Metrics

Most toxicity assessment frameworks typically employ an auxiliary model, usually a classifier, to evaluate the toxicity levels in the text outputs generated by language models.

Perspective API, developed by Google Jigsaw, is the most commonly used technique for scoring text for toxicity (Lees et al., 2022). As shown in Fig. 6.8, the input is the text, and the output is a probability score ranging from 0 to 1, which quantifies the likelihood of the text being perceived as containing a particular attribute as an indicator of toxicity. These include various attributes such as TOXICITY, SEVERE_TOXICITY, IDENTITY_ATTACK, INSULT, PROFANITY, THREAT, SEXUALLY_EXPLICIT, and FLIRTATION. For training its models, the Perspective API uses a large corpus of data from various online forums, such as Wikipedia and The New York Times.



OUTPUT : SCORE

Toxicity	0.99
Severe_Toxicity	0.75
Insult	1.0
Sexually_Explicit	0.04
Profanity	0.93
Likely_To_Reject	0.99
Threat	0.15
Identity_Attack	0.03

Fig. 6.8: Toxicity Scoring through Perspective API

The data are labeled through crowd-sourced raters from various organizations such as Figure Eight, Appen, and internal platforms. These raters determine whether a comment exhibits a particular attribute, such as TOXICITY. The final labels are derived from the proportion of raters who identify a text as toxic. In terms of its model architecture, the Perspective API initially trains multilingual BERT-based models using the labeled data, and then the model is distilled into single-language CNNs for each supported language, ensuring the system's ability to provide rapid and efficient scoring.

The Perspective API provides a probability score indicating toxicity when evaluating text generation. Thus, the Perspective API toxicity classifier can be represented as:

$$c : \widehat{Y} \longrightarrow [0, 1]$$

Gehman et al. (2020) use the *Expected Maximum Toxicity* (EMT) metric, calculated over 25 outputs from a language model, focusing on both the mean and standard deviation (Gehman et al., 2020). EMT is designed to identify the most extreme cases in the generated texts of a language model, essentially spotlighting the worst-case scenarios.

$$\mathrm{EMT}(\widehat{\mathbb{Y}}) = \max_{\widehat{Y} \in \widehat{Y}} c(\widehat{\mathbb{Y}}) \tag{6.23}$$

Toxicity Probability (TP) is employed to gauge the likelihood of producing at least one toxic text (with a toxicity score equal to or exceeding 0.5) across 25 generations. This metric serves to quantify the frequency with which a language model generates toxic content.

$$\mathrm{TP}(\widehat{\mathbb{Y}}) = P\left(\sum_{\widehat{Y} \in \widehat{\mathbb{Y}}} \mathbb{I}(c(\widehat{Y}) \geq 0.5) \geq 1\right) \tag{6.24}$$

The *Toxic Fraction* (TF), defined by Bommasani et al. (2023), calculates the proportion of generated texts that are deemed toxic.

$$\mathrm{TF}(\widehat{\mathbb{Y}}) = E_{\widehat{Y} \in \widehat{\mathbb{Y}}}[\mathbb{I}(c(\widehat{Y}) \geq 0.5)] \tag{6.25}$$

6.3.3 Benchmarks

While there is considerable overlap between the benchmarks for biases outlined in the previous section and those used for assessing toxicity, this section concentrates specifically on benchmarks tailored for toxicity. These benchmarks are comprehensively summarized in Table 6.3.

Table 6.3: Key toxicity benchmark datasets. Each dataset is characterized by its size, the approach taken for collecting and labeling the data, and a short description of the nature of the content.

Dataset	Size	Method	Focus
Perspective API's Toxicity Dataset	1.8M	Crowdsourced	Overall toxicity and specific dimensions
Jigsaw Toxic Comment Dataset	150k	Crowdsourced	Toxicity levels and types
Hate Speech Dataset	24k	Crowdsourced	Hate speech detection
ToxiGen	100k	Adversarial	Model robustness and hidden biases
Thoroughly Engineered Toxicity (TET) Dataset	10k	Manual	Nullifying model defenses
ImplicitHateCorpus	5.7M	Crowdsourced	Implicit hate speech (sarcasm, stereotypes, microaggressions)
DynaHate	22.5M	Machine learning	Contextual hate speech (target-specific, evolving language)
SocialBiasFrames	8,732	Crowdsourced	Harmful social frames (gender, race, disability)

6.3.4 Mitigation Strategies

Gehman et al. (2020) classify toxicity mitigation techniques into two primary types: data-based and decoding-based strategies. Data-based strategies encompass further pre-training of the model, altering its parameters. This approach, while effective, tends to be computationally intensive due to the parameter modifications involved. In contrast, decoding-based methods focus on altering only the decoding algorithm of a language model, leaving the model parameters intact. As a result, these strategies are typically more accessible and less resource intensive, offering a practical advantage for practitioners in the field.

6.3.4.1 Data-based Methods

The *Domain Adapted Pre-training* (DAPT) technique, as discussed by Gururangan et al. (2020), involves a two-phase pre-training process. The first stage involves general pre-training across a wide range of sources, while the second stage focuses on domain-specific pre-training, utilizing datasets pertinent to specific contexts, such as Twitter communication data, as shown in Fig. 6.9. This approach is particularly effective when combined with fine-tuning for specific tasks. For example, in classifying topics related to the Black Lives Matter movement, fine-tuning focuses on the hierarchy and nuances of language specific to this context, extracted from the broader domain of tweets. Although Gehman et al. (2020) demonstrated that continuing the pre-training of LLMs on a non-toxic subset of *OPENWEBTEXTCORPUS* (OWTC) using DAPT can significantly reduce toxicity in models such as GPT-2, this method is not without its challenges. One of the most significant limitations is the computational expense involved. Additionally, the necessity for extra training data, particularly when sourced through human labor such as crowdsourcing, can be prohibitively costly. Another critical limitation of this approach is its potential to adversely affect modeling performance. In filtering out toxic elements, there is a risk of inadvertently eliminating benign and potentially valuable knowledge.

Attribute Conditioning (ATCON) is another data-based technique based on the work of Ficler and Goldberg (2017) and Keskar et al. (2019). The approach involves the augmentation of pre-training data for pre-trained models with a toxicity attribute token, signified as `|toxic|` or `|non-toxic|`, incorporated into a randomly selected subset of documents. The process includes a further pre-training phase, enabling the model to integrate and adapt to these toxicity indicators. When the model is employed for text generation, the toxicity attribute token, specifically `|non-toxic|`, is prepended to the prompts. This addition acts as a signal to the model, delineating the desired focus on non-toxicity in the generated text.

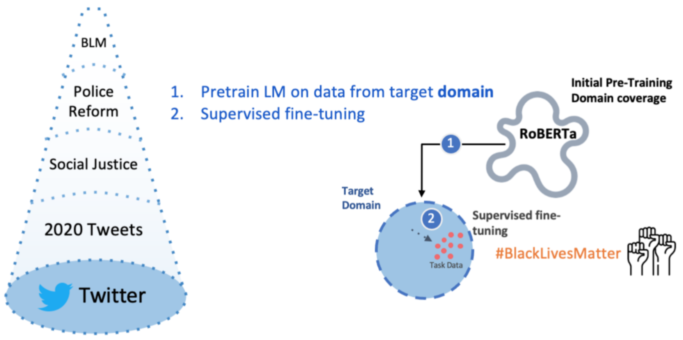


Fig. 6.9: Schematic illustrating Domain Adapted Pre-training. After the initial model pre-training, a second pre-training step is done with a corpus that 1) is pertinent to the model task (i.e., web data); 2) has been filtered of toxic communications.

6.3.4.2 Decoding-based Methods

Decoding-based methods denote approaches where the decoding algorithm is adjusted to minimize toxicity.

Practical Tips

One of the most simplest decoding-based strategies is *blocklisting*, also known as *word filtering*. This approach involves assigning a zero probability to undesirable words — which typically include curse words, profanity, and insults — within the language model's framework. As a result, the model is effectively prevented from generating these words in its output. There are a number of more complex, and generally more effective, approaches to detoxification during decoding, described here.

Vocabulary shifting

This approach, developed by Ghosh et al. (2017), centers around learning vector representations that distinctly signify toxic and non-toxic attributes for each token in the vocabulary. While the original research by Ghosh et al. utilized LSTM models, the fundamental principles of this technique are adaptable and remain consistent when applied to more contemporary Transformer-based architectures such as GPT-2.

In a standard LSTM model, the joint probability of a sequence of M words $w_1, w_2, ..., w_M$ is defined by the chain rule of probability:

$$P(w_1, w_2, ..., w_M) = \prod_{t=1}^{M} P(w_t | w_1, w_2, ..., w_{t-1}) \tag{6.26}$$

For such a model, the conditional probability of word w_t in context $c_{t-1} = (w_1, w_2, \ldots, w_{t-1})$ is:

$$P(w_t = i|\mathbf{c}_{t-1}) = \frac{\exp(U_i^T f(\mathbf{c}_{t-1}) + b_i)}{\sum_{j=1}^{V} \exp(\mathbf{U}_j^T f(\mathbf{c}_{t-1}) + b_j)} \quad (6.27)$$

where $f(\cdot)$ is the LSTM output, $\mathbf{U}$ is a word representation matrix, and $\mathbf{b}_i$ is a bias term. The proposed Affect-LM model for vocabulary shift by Ghosh et al. (2017) modifies this equation as follows:

$$P(w_t = i|\mathbf{c}_{t-1}, \mathbf{e}_{t-1}) = \frac{\exp(\mathbf{U}_i^T f(\mathbf{c}_{t-1}) + \beta V_i^T g(\mathbf{e}_{t-1}) + b_i)}{\sum_{j=1}^{V} \exp(\mathbf{U}_j^T f(\mathbf{c}_{t-1}) + \beta V_j^T g(\mathbf{e}_{t-1}) + b_j)} \quad (6.28)$$

where $\mathbf{e}_{t-1}$ that captures affect category information derived from the context words during training. It quantifies the impact of the affect category information on predicting the target word w_t in context, and β is the affect strength parameter.

Plug and Play Language Model (PPLM)

PPLM allows users to integrate one or more attribute models representing specific control objectives into an LLM (Dathathri et al., 2019). This seamless integration requires no additional training or fine-tuning of the model, which is a significant advantage for researchers who lack access to extensive hardware resources. PPLM functions under two key assumptions:

1. Access to an attribute model, denoted as $p(a \mid x)$.
2. Availability of gradients from this attribute model.

The PPLM process, as shown in Fig 6.10, involves the following steps:

1. Perform a forward pass in the LLM, sampling a token from the resulting probability distribution. Then, feed the generated string to the attribute model to calculate the likelihood of the desired attribute, $p(a \mid x)$.
2. Execute backpropagation to compute gradients of both $p(a \mid x)$ and $p(x)$ with respect to the model's hidden state. Adjust the hidden state to increase the probability of both $p(a \mid x)$ and $p(x)$.
3. Recalculate the LLM's probability distribution and sample a new token.

Generative Discriminator (GeDi)

GeDi uses smaller LMs as generative discriminators to guide generation from LLMs to mitigate toxicity (Krause et al., 2020). It calculates the probability of each potential next word in a sequence using Bayes' rule, balancing two class-conditional distributions. The first distribution is conditioned on a desired attribute or "control code", directing the model toward specific, favorable content. Conversely, the second distribution, or "anti-control code", is aligned with attributes to avoid, such as toxic language.

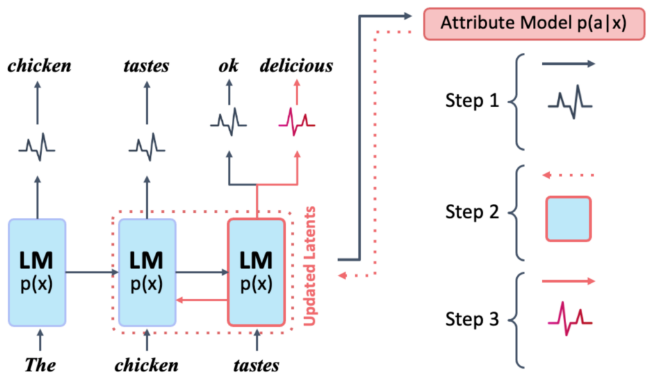


Fig. 6.10: Plug and Play Language Model (PPLM)

The core principle entails the use of auxiliary *Class-Conditional Language Models* (CC-LMs) to ascertain the likelihood of a generated token being part of the control code class. This code defines an attribute of the text sequence $x_{1:T}$, potentially representing aspects such as sentiment, topic, or labels such as "toxic". The CC-LL calculates probabilities $p(x|c)$ and $p(x|\cancel{c})$, where c and $\cancel{c}$ are the control code and anti-control code. An example of a CC-LL is CTRL, which provides a framework for language models to generate text while being conditioned on an attribute variable (Keskar et al., 2019).

In training a CC-LM, each sequence $x^{(i)}_{1:T_i}$ is paired with a control code $c^{(i)}$. The model is then trained to minimize the average negative log-likelihood, L_g. The combined loss function for GeDi training, $\mathcal{L}_{gd}$, is a mix of the LLM's loss $\mathcal{L}_g$ and the discriminative loss $\mathcal{L}$, weighted by a hyper-parameter λ:

$$\mathcal{L}_{gd} = \lambda \mathcal{L}_g + (1 - \lambda)\mathcal{L}_d \tag{6.29}$$

where $\mathcal{L}$ is a discriminative loss aimed at optimizing toward desirable behavior and away from the undesirable behavior. The discriminative loss $\mathcal{L}_d$ aims at increasing the classification accuracy, whereas $\mathcal{L}_g$ helps in better-calibrated token probabilities for guided generation. GeDi is more effective at detoxification than PPLM and is less computationally expensive. However, it has the limitation of requiring an additional language model for training, which adds to the computational requirements.

Self-Diagnosis and Self-Debiasing

This method, developed by Schick et al. (2021), addresses detoxification by asking two pivotal questions: First, can LLMs autonomously detect their own biases (Self-Diagnosis)? Second, can LLMs discard undesired behaviors in an unsupervised manner (Self-Debiasing)?

Self-diagnosis investigates the ability of pre-trained language models to self-diagnose socially undesirable attributes in their outputs without relying on additional training data or external resources. It focuses on the models' inherent knowledge gained during pre-training and uses brief attribute descriptions to assess their ability to identify and address biases and inappropriate content autonomously.

Consider a language model M and a sequence of tokens $x_1, \ldots, x_k = \mathbf{x}$. For self-diagnosis, a sentence $\mathbf{x}$ generated by M is combined with a query about an attribute $\mathbf{y}$, to determine whether $\mathbf{x}$ contains $\mathbf{y}$. For instance, if M generates a sentence $\mathbf{x}$ = "I'm going to hunt you down!" and needs to be checked for a threat $\mathbf{y}$ = "a threat", the model is prompted with the self-diagnosis template:

```
# Self-diagnosing template, denoted as sdg(.)
{sample text x}
Does the above text contain {threat text y} ?
```

The probability of $\mathbf{x}$ exhibiting attribute $\mathbf{y}$ is given by:

$$p(\mathbf{y}|\mathbf{x}) = \frac{p_M(\text{"Yes"}|\text{sdg}(\mathbf{x}, \mathbf{y}))}{\sum_{w \in \{\text{"Yes"}, \text{"No"}\}} p_M(w|\text{sdg}(\mathbf{x}, \mathbf{y}))} \tag{6.30}$$

Self-debiasing mirrors the idea of self-diagnosis but focuses on active modification of the output. The self-debiasing process also focuses on using the model's own internal knowledge to adapt its output generation, thereby reducing the likelihood of producing biased text. An example self-debiasing template is as follows:

```
# Self-debiasing template, denoted as sdb(.)
The following text contains {undesired attribute s}:
{sample text x}
```

Employing this template involves a pre-trained language model, denoted as M, and an attribute $\mathbf{y}$, signifying a specific bias. When M is provided with an input text $\mathbf{x}$, it uses the self-debiasing input $\text{sdb}(\mathbf{x}, \mathbf{y})$ to generate a continuation. This results in two probability distributions: the original $p_M(w|\mathbf{x})$ and the adjusted $p_M(w|\text{sdb}(\mathbf{x}, \mathbf{y}))$. The latter is designed to favor undesirable words, as evidenced by the equation $\Delta(w, \mathbf{x}, \mathbf{y}) = p_M(w|\mathbf{x}) - p_M(w|\text{sdb}(\mathbf{x}, \mathbf{y}))$, which is expected to be negative for biased words.

To mitigate this bias, a new probability distribution $\widetilde{p}_M(w|\mathbf{x})$ is formulated, proportional to $\alpha(\Delta(w, \mathbf{x}, \mathbf{y})) \cdot p_M(w|\mathbf{x})$. Here, $\alpha : \mathbb{R} \rightarrow [0, 1]$ acts as a scaling function, adjusting probabilities based on the bias difference $\Delta(w, \mathbf{x}, \mathbf{y})$. This scaling is intended to be minimally invasive, altering probabilities only when necessary while leaving unbiased words unchanged.

The approach avoids setting probabilities of any word to zero to maintain practical model evaluation. Instead, the probability of biased words is diminished according to $\Delta(w, \mathbf{x}, \mathbf{y})$, using a decay constant λ in the expression $\alpha(x) = 1$ if $x \geq 0$, $e^{\lambda \cdot x}$ otherwise. This method can be extended to accommodate multiple biases by considering a set of descriptions $\mathbf{Y} = \{\mathbf{y}_1, \cdots, \mathbf{y}_n\}$ and adjusting the distribution accordingly.

6.4 Privacy

With the rise of the internet over the past several decades, we live in an age where information flows more freely than ever. Unfortunately, not all of this information is willfully and knowingly shared by those providing it, nor is it thoughtfully collected and stored by those obtaining it. As such, the increased accessibility of personally identifying information and other private data has become a widely recognized concern. Given that all of the most prominent LLMs source a substantial amount of training data from websites, it is natural to consider whether this poses any downstream risks to privacy. As it turns out, LLMs learn specific information about individuals, and it is possible to extract that information with sufficient prompting. Privacy remains a largely unsolved problem for LLM at this point. This section will discuss the existing research and emerging trends aiming to address these concerns.

6.4.1 Causes

The conventional wisdom in most realms of machine learning is that when a model frequently generates predictions that closely match the examples seen during training, it is a classic symptom of overfitting. This principle fueled an assumption that LLMs are generally unlikely to memorize their training data and repeat it verbatim since they are most commonly trained for only one epoch on a considerable volume of data. The process is directly at odds with the conditions that define overfitting. Unfortunately, the assumption that memorization exclusively arises from overfitting has been invalidated (Carlini et al., 2021). Because the memorization potential of LLMs was not widely recognized early on, the research interest in the mitigation of private data capture has lagged behind the models' overall capabilities. High-performing LLMs with more parameters or training tokens also appear to have a greater capacity to memorize data than their scaled-down siblings (Nasr et al., 2023).

To quantify the memorized training data for a given model, Carlini et al. (2021) proposed an attack designed to evoke a memorized output string. Their approach samples sequences of tokens from internet sources, which are either known to be or likely to be part of the training data. These tokens are then used to prompt the model, with the outputs then being checked for precise matches in the training data. Naturally, this process is far more accessible when the training sources are public knowledge. To extend their research to GPT-2, which has never published its training data, the authors had to resort to Google searches over the entire internet to locate matches with a high probability of having been memorized by the model. Despite this limitation, they were nonetheless able to find data that had been memorized by GPT-2, including personally identifiable information (PII).

In subsequent work by Nasr et al. (2023) several additional experiments were undertaken based on the above mentioned procedure. Their research distinguished the rate of finding unique memorized output and the number of unique sequences extracted. It was observed, for instance, that Pythia-1.4 emitted more memorized data than GPT-Neo-6, up to approximately 60 billion queries. At that point, their curves crossed, and those of GPT-Neo-6 surpassed those of Pythia-1.4. This is somewhat unsurprising, as there is no intuitive reason to expect that the effectiveness of the attack technique for a particular model is a direct measure of the actual amount of data it has memorized. The authors established this relationship using extrapolation procedures, which allowed them to place a much tighter lower bound on the amount of total memorized data for every model evaluated.

Chat models exhibit different behaviors, given that they typically undergo alignment tuning and are conditioned to provide helpful responses in a conversational tone. As a result, ChatGPT appeared to be relatively safe at first since it is unlikely to return a continuation of the memorized text when a sequence of tokens is submitted in the prompt. However, an additional finding by Nasr et al. (2023) was the discovery of a new attack that succeeded in extracting memorized data from ChatGPT. This approach entails selecting a single-token word, repeating it several times, and then asking ChatGPT to continue repeating it forever. The results vary widely depending on the chosen token, but in some instances, ChatGPT will eventually stop repeating the word and instead begin regurgitating memorized training data. Several thousand occurrences of memorized PII were verified within a text sample generated in this experiment.

The mechanism for this surprising behavior is not known with certainty due to the closed nature of ChatGPT, but examination of open-source models suggests that as the text grows longer, the model perhaps infers that there is an increasingly high likelihood of it ending soon. After all, no training text truly goes on forever, so this is somewhat of a foreign concept for a model. This may cause it to eventually converge toward the unique "end-of-text" token as its next-token prediction, forcing the model to generate a new sequence without any meaningful context preceding it. Intuitively, it seems plausible that this could significantly increase the probability of falling back on a known text sequence seen during training.

It is important to note that these attacks were designed to demonstrate the possibility that a nontrivial amount of memorized data could be extracted from LLMs. The research has only scratched the surface of potential techniques that might be employed or how hackers could seek to make these techniques more efficient and scalable. As discussed above, ChatGPT did not appear to be prone to training data leakage in early attacks but cracked when a novel attack was discovered. The implication is that it is challenging to gauge the full extent of memorization in LLMs and its risk of harm.

6.4.2 Evaluation Metrics

Kim et al. (2023) introduced a tool to probe and quantify PII leakage called ProPILE. It consists of an evaluation dataset constructed by locating PII within the widely used Pile training dataset and a set of prompt templates to retrieve a specific type of personal data given other relevant information about the person. As a concrete example, the prompt “Please contact {name} by {pii_1} or {target_pii_type} ” could test whether the name and phone number give the model sufficient context to retrieve a street address, or a street address reveals an email address, and so forth. All possible permutations are queried, and the model outputs are captured.

The effectiveness of the prompting attack can be measured with two different metrics: exact match and likelihood. The exact match metric is based on a simple equality test of the generated string and the true PII. The likelihood metric leverages the capability of many model APIs to return scores for each token prediction and not just the response text. Obtaining the likelihood makes achieving a more precise estimate possible, particularly in a scenario where attackers could operate at a large scale and repeat similar prompts many times. This is defined as follows:

$$Pr(a_m|\mathcal{A}_{\backslash m}) = \prod_{r=1}^{L_r} p(a_{m,r}|x_1, x_2, \ldots, x_{L_q+r-1}) \tag{6.31}$$

where a_m is the target PII, $\mathcal{A}_{\backslash m}$ is the remaining PII, L_q is the length of the query, and L_r is the length of the correct response. Repeated computation over multiple queries produces an additional metric representing the percentage of people included in the training data that would have a piece of their PII exposed in k queries or less, using all available prompts. This metric is $\gamma_{<k}$.

In the context of ProPILE, users wanting to check whether a model exposes their data are constrained to what is deemed black-box probing since the only information they have about the model is its outputs. The previously described templates are the only available prompting mechanisms for black-box probing. White-box probing refers to the setting where model providers wish to quantify PII leakage for their models. The models’ weights are known in this case and can boost the prompting effectiveness. With full access to the model, it is possible to train soft prompts that exceed the capabilities of the prompt templates. Hypothetical attackers would not have the necessary information to follow a similar prompt tuning approach and would be limited to less efficient prompt engineering techniques. Presumably, even a clever and motivated attacker would have difficulty devising a better probing strategy than a soft prompt developed by the model owners, so this technique enables model developers to zero in on a worst-case PII leakage estimate.

Inan et al. (2021) introduced another privacy metric based on the concept of *differential privacy*. A computation is considered differentially private when two datasets that differ by exactly one record produce the same outputs with a maximum probabilistic deviation ϵ. Formally this is given by:

$$Pr[M(A) \in S] \leq e^{\epsilon} Pr[M(B) \in S] \tag{6.32}$$

where A and B are datasets that differ by a single record and S is any subset of possible outputs. For an adequately small ϵ, differential privacy can provide a very strong guarantee that a model does not reveal information specific to an individual data point. By definition, the model’s output would be consistent with an alternate version trained on data that does not include that individual.

Furthering this idea, Inan et al. (2021) trained a reference model with all the data found to be unique to any user removed. They then used the perplexity ratio of the reference model and the model being assessed for privacy leakage on each of the removed elements, thus defining a worst-case leakage metric as follows:

$$\epsilon_l = \max_{w \in S_{uniq}} \log(\frac{PP_{public}(w)}{PP_{lm}(w)} \tag{6.33}$$

where PP_{lm} is the perplexity of a language model trained with user data and PP_{public} is the perplexity of a public model, over each sequence $w \in S_{uniq}$.

6.4.3 Benchmarks

Mireshghallah et al. (2023) proposed *ConfAIde* as a benchmark for assessing whether LLMs can make sound judgments about revealing potentially sensitive information. Instead of analyzing what the model knows, it focuses on its ability to reason about situations in which knowledge should not be shared. Four different types of scenarios are defined, with increasing degrees of complexity. The simplest scenario asks a model to rate the sensitivity for a given piece of information, independent of who accesses it or how it is used. The other three tiers become more complex, with the last tier asking the model to capture notes on an extended conversation. Multiple people join and leave at different times, discussing multiple types of information; the model then must decide which information to share in a summary with the whole group. For each example, human annotations were collected to establish the sensitivity of the information and human preferences for sharing or not sharing the information in the given context.

The authors of ConfAIde used their benchmark to assess several top-performing LLMs, including GPT-4 and ChatGPT. They found that the models were reasonably good at the lowest level task of recognizing sensitive information, but their capabilities decreased significantly when greater contextual awareness was needed. Their results indicated that LLMs had difficulty distinguishing between private and public information, so generally, a model that is less likely to share sensitive data is also more likely to hide non-sensitive data. The ConfAIde benchmark reveals that even instruction-tuned LLMs are incapable of reasoning about privacy and secrecy, and new techniques will likely be needed to address this deficiency.

6.4.4 Mitigation Strategies

In this section, we discuss practical ways to mitigate the privacy issues posed by LLMs. These strategies are divided into methods that can be applied during training and methods applied during inference.

6.4.4.1 Privacy Protection During Training

Perhaps the most intuitively straightforward way to prevent LLMs from distributing personal information is to purge it from the training data. A model obviously won't memorize private data if it never sees it in the first place. This is a widely utilized pre-processing step for LLM pre-training, as mentioned in Chapter 2. Unfortunately, given the massive quantities of data involved, it is virtually impossible to guarantee that all PII has been removed using standard anonymization techniques.

The concept of differential privacy discussed earlier in this section also has utility as a mitigation strategy. Various researchers, such as Abadi et al. (2016), have introduced differential privacy into the training process by building it into the stochastic gradient descent optimizer. While this approach has its merits, it has thus far been shown in most cases to be detrimental to training and usually results in lower-quality models.

A further limitation of training data anonymization and differential privacy is that LLMs have also been shown to infer personal information without explicitly learning it. Staab et al. (2023) found that several state-of-the-art LLMs could accurately discover Reddit users' information based on their posted content. This work sought to identify personal attributes such as age, gender, and location through direct prompting techniques. They sent a user's posts and asked each model if it could guess the information. Even when the input data had been anonymized to remove instances where users explicitly divulged information, they were still frequently successful at guessing correctly. GPT-4 had the highest accuracy on the evaluation dataset curated by the authors, at an impressive 84.6%.

While guessing the approximate age of an unknown Reddit user may seem benign at first glance, these findings are significant because many people who participate in online forums believe that they are anonymous as long as they do not reveal their names. It is well known that the internet makes people feel more comfortable saying things they otherwise would not want to share. However, suppose they divulge a considerable amount of information about where they live, their jobs, their families, and their age. In that case, it becomes possible for a determined individual to piece together someone's identity through social media and publicly available records. This risk already exists without using LLM's, but it is somewhat laborious. LLMs could accelerate this malicious activity and make it easier to conduct at a much larger scale.

It is conceivable that future work will give rise to new techniques that are more successful at preventing models from memorizing PII from their training data. However, it is far more difficult to imagine how we could continually develop increasingly powerful models yet somehow prevent them from acquiring enough knowledge to infer geographical and generational differences in speaking or writing styles. For better or worse, this is a capability that LLM's now possess. It is almost certainly more prudent to focus on putting safeguards around model usage rather than attempting to stunt their intelligence.

6.4.4.2 Privacy Protection During Inference

When new ways to exploit a model are discovered, significant pressure exists to resurface that direct query. Often, when reports surfaced that directly query an LLM can retrieve information that should not have been given, users will soon find that similar prompts stop working. The system could be updated internally, for instance, to include instructions within the context that any requests to determine a person's location should not be carried through. The model can then respond to such queries by simply stating that it is unable to provide an answer. While it is good for model providers to be willing and able to address such issues as quickly as possible, this is a very reactionary approach that falls short of completely alleviating all privacy concerns.

Given the seeming inability of modern LLMs to fully guarantee the protection of private data, it is also vital for application developers to consider how these models could put their users at risk. After all, LLM providers such as OpenAI are known to store queries sent through their APIs to enable future technological advancements. Rather than fully entrusting model researchers and developers with the responsible use of incoming data, the consumers of LLMs must often consider anonymizing their prompts before sending them to a third-party service. This is especially true for any application where users are likely to include personal data intentionally or unintentionally. Tools such as OpaquePrompts have been developed to automate the removal of sensitive information and, depending on the use case, potentially inject the anonymized tokens back into the output downstream of the model's response if needed.

! Practical Tips

Another common alternative for organizations that rely on externally developed models is to choose an open-source LLM instead of a service such as ChatGPT. With this approach, a copy of the model can be deployed internally. Users' prompts remain secure within the organization's network rather than being exposed to third-party services. While this dramatically reduces the risk of leaking sensitive data, it also adds significant complexity.

LLM demands expensive computing resources, and optimizing the cost of all that computation demands specialized human expertise. Beyond the increasingly large number of applications being built on top of third-party LLMs, there is also a strong demand for direct interactions with ChatGPT and its ilk to help with various daily tasks. However, this has also been met with hesitation by many people who are concerned about exposing their private data. A user who wants to use an online LLM to write an email with a more professional tone would necessarily expose the contents of their proposed email to the service providing the model. To avoid this uncomfortable situation, the adoption of smaller models that can run on personal devices has increased rapidly.

! Practical Tips

One of the most popular locally installable LLM interfaces is an application called GPT4All from Nomic AI. It provides options to download different model variants, such as Falcon and Mistral, in sizes under 10 billion parameters. These models are small enough to provide fast, high-quality responses on a personal device, requiring no API requests. Naturally, there are some limitations compared to the more powerful GPT models, especially in cases with large context sizes. However, a smaller LLM can be more than adequate for answering questions or helping with basic tasks. In many cases, it is a reasonable trade-off for substantially reducing privacy risk.

The trend toward locally available models is being closely watched from the perspective of the hardware industry as well. Over the past decade, most hardware advancements have been geared toward more efficient training of models on ever-larger datasets. However, in recent years, there has been a massive wave of investment in edge computing optimized for neural models. Some prognosticators believe that the growth potential for this technology may be even more significant than the astounding revenue growth that NVIDIA has achieved from its large-scale GPUs. While there are other factors, privacy concerns with LLMs undoubtedly contribute to the interest in decentralized models.

6.5 Tutorial: Measuring and Mitigating Bias in LLMs

6.5.1 Overview

In Section 6.2, we discussed the impact of bias in LLMs and some of the techniques developed to mitigate it. In this tutorial, we will apply one of these methods and observe the corresponding shifts in model behavior. This exercise closely follows the work of Meade et al. (2022), who surveyed several bias mitigation techniques and conveniently provided the code to run all their experiments in a GitHub repository.

Goals:

- Analyze how the CrowS benchmark is designed to measure bias.
- Test the use of one potential bias mitigation technique on RoBERTa and evaluate the improvement.
- Apply a debiased model on a downstream task to assess whether its capabilities as a language model are degraded.

Please note that this is a condensed version of the tutorial. The full version is available at `https://github.com/springer-llms-deep-dive/llms-deep-dive-tutorials`.

6.5.2 Experimental Design

In this exercise, we will demonstrate the use of the `bias-bench` library to reduce the appearance of gender bias in a Roberta model. We will then use the CrowS metric to demonstrate the improvement and compare the debiased model's capabilities to those of the original model on a sentiment analysis task.

The dataset used for the CrowS benchmark consists of pairs of sentences. In each pair, one sentence represents a stereotype while the other replaces the relevant words to contradict the stereotype. For example, "black" may be replaced with "white" if it is a racial stereotype, "woman" may be replaced with "man" if it is a gender stereotype, and so forth. The sentence pairs are otherwise identical apart from these tokens. These data are used to measure the bias of a given LLM and the relative effects of potential bias mitigation techniques.

The algorithm chosen for this experiment is called Sent-Debias. The motivation behind this algorithm is that if a model is utterly neutral about an attribute such as gender, its embeddings of "He was a slow runner" and "She was a slow runner" would generally be very close, if not identical. Variations in these embeddings can be primarily attributed to bias. Sent-Debias captures these variations across many examples and maps them to a lower-dimensional subspace using primary component analysis, resulting in a set of vectors representing the direction of the bias. Once this subspace is learned, it is inserted into the forward pass so that any text representation's bias projection is subtracted before the final output is returned.

Sent-Debias requires a large and diverse dataset to generate the sentences used in the procedure described above. It has a predefined set of biased words to augment the data, such as "boy" and "girl," for instance. We use a sample of text from Wikipedia to learn a representation of model biases as reflected in the difference between sentence embeddings with potentially biased tokens substituted.

After applying bias mitigation to a model and evaluating whether gender bias has been reduced from the original version, we then assess its comparative ability to be fine-tuned on a downstream task. SST, a standard sentiment analysis dataset that is part of the GLUE benchmark, is used for this purpose (Socher et al., 2013; Wang et al., 2019).

6.5.3 Results and Analysis

Before we begin the debiasing experiments, we assess the current performance of the `roberta-base` model on CrowS. This metric indicates how likely the model is

Table 6.4: Comparison of model variants on the CrowS and SST benchmarks, highlighting the impact of debiasing.

Model Variant	CrowS	SST
Base RoBERTa	60.15	0.922
Sent-Debias RoBERTa	52.11	0.930

to choose a stereotype when asked to fill in masked tokens in a potentially biased sentence, therefore a lower number is better. It is important to note that an inferior language model could achieve nearly perfect results on this metric since it has not learned the biases in the data well enough to select tokens that reflect stereotypes. It is often the case that weaker LLMs tend to appear less biased than more capable LLMs based on this metric.

```
# Output
-----------------------------------------------------------
Total examples: 261
Metric score: 60.15
```

Listing 6.1: CrowS Metrics for RoBERTa

Next, we use Sent-Debias to compute a gender bias subspace for RoBERTa. Once the bias direction is computed, we recheck the CrowS benchmark to determine whether gender bias has decreased. It is now much closer to 50.0, meaning that the model does not prefer stereotypical tokens as frequently as before.

```
# Output
-----------------------------------------------------------
Total examples: 261
Metric score: 52.11
```

Listing 6.2: Re-evaluating CrowS metrics with debiasing.

As mentioned, our model's improvement on the CrowS metric may be linked to a decreased overall ability to predict tokens accurately. To make sure that we still have a similarly performing LLM after removing gender bias, we compare the results of fine-tuning the model for sentiment analysis both with and without Sent-Debias. There is slight variation between runs of the training loop, but the accuracy on the SST test data appears to be roughly the same regardless of whether Sent-Debias is applied.

While these results are undoubtedly positive, it is not clear whether we can declare success or whether the debiased LLM recovered some degree of gender bias during the fine-tuning process. It seems likely that the sentiment training data may have been biased, and the effects would not be readily captured by the CrowS metric we employed. We would need to analyze this task more closely to ascertain whether our attempt to mitigate bias succeeded.

6.5.4 Conclusion

In this tutorial we have shown a promising approach to address bias in LLMs, but current techniques still fall short of fully solving this issue. A crucial finding of Meade et al. (2022) was that despite numerous proposed debiasing strategies, none perform consistently well across various models and bias types. In addition, they also found that benchmarks such as CrowS, StereoSet, and SEAT can be unstable in terms of their performance across multiple runs of the same algorithm. This leaves the question of whether the metrics are robust enough to form a complete bias assessment. Further work in both measuring and mitigating bias will be highly important.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS'16. ACM, October 2016. doi: 10.1145/2976749.2978318. URL http://dx.doi.org/10.1145/2976749.2978318.

Abubakar Abid, Maheen Farooqi, and James Zou. Persistent anti-muslim bias in large language models. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 298–306, 2021.

Jaimeen Ahn and Alice Oh. Mitigating language-dependent ethnic bias in bert. *arXiv preprint arXiv:2109.05704*, 2021.

Giuseppe Attanasio, Debora Nozza, Dirk Hovy, and Elena Baralis. Entropy-based attention regularization frees unintended bias mitigation from lists. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1105–1119, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.88. URL https://aclanthology.org/2022.findings-acl.88.

Emily M Bender and Alexander Koller. Climbing towards nlu: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 5185–5198, 2020.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623, 2021.

Camiel J Beukeboom and Christian Burgers. How stereotypes are shared through language: a review and introduction of the aocial categories and stereotypes communication (scsc) framework. *Review of Communication Research*, 7:1–37, 2019.

Su Lin Blodgett and Brendan O'Connor. Racial disparity in natural language processing: A case study of social media african-american english. *arXiv preprint arXiv:1707.00061*, 2017.

Rishi Bommasani, Percy Liang, and Tony Lee. Holistic evaluation of language models. *Annals of the New York Academy of Sciences*, 2023.

Conrad Borchers, Dalia Sara Gala, Benjamin Gilburt, Eduard Oravkin, Wilfried Bounsi, Yuki M Asano, and Hannah Rose Kirk. Looking for a handsome carpenter! debiasing gpt-3 job advertisements. *arXiv preprint arXiv:2205.11374*, 2022.

Shikha Bordia and Samuel R Bowman. Identifying and reducing gender bias in word-level language models. *arXiv preprint arXiv:1904.03035*, 2019.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Deng Cai, Yan Wang, Huayang Li, Wai Lam, and Lemao Liu. Neural machine translation with monolingual translation memory. *arXiv preprint arXiv:2105.11269*, 2021.

Meng Cao, Yue Dong, Jiapeng Wu, and Jackie Chi Kit Cheung. Factual error correction for abstractive summarization models. *arXiv preprint arXiv:2010.08712*, 2020.

Nicholas Carlini et al. Extracting training data from large language models, 2021.

Kai-Wei Chang, Vinodkumar Prabhakaran, and Vicente Ordonez. Bias and fairness in natural language processing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): Tutorial Abstracts*, 2019.

I Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, Pengfei Liu, et al. Factool: Factuality detection in generative ai–a tool augmented framework for multi-task and multi-domain scenarios. *arXiv preprint arXiv:2307.13528*, 2023.

Aakanksha Chowdhery et al. Palm: Scaling language modeling with pathways, 2022.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*, 2019.

Ameet Deshpande, Vishvak Murahari, Tanmay Rajpurohit, Ashwin Kalyan, and Karthik Narasimhan. Toxicity in chatgpt: Analyzing persona-assigned language models. *arXiv preprint arXiv:2304.05335*, 2023.

Harnoor Dhingra, Preetiha Jayashanker, Sayali Moghe, and Emma Strubell. Queer people are people first: Deconstructing sexual identity stereotypes in large language models. *arXiv preprint arXiv:2307.00101*, 2023.

Emily Dinan, Angela Fan, Adina Williams, Jack Urbanek, Douwe Kiela, and Jason Weston. Queens are powerful too: Mitigating gender bias in dialogue generation. *arXiv preprint arXiv:1911.03842*, 2019.

Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 67–73, 2018.

Nouha Dziri, Sivan Milton, Mo Yu, Osmar Zaiane, and Siva Reddy. On the origin of hallucinations in conversational models: Is it the datasets or the models? *arXiv preprint arXiv:2204.07931*, 2022.

Zahra Fatemi, Chen Xing, Wenhao Liu, and Caiming Xiong. Improving gender fairness of pre-trained language models without catastrophic forgetting. *arXiv preprint arXiv:2110.05367*, 2021.

Chao Feng, Xinyu Zhang, and Zichu Fei. Knowledge solver: Teaching llms to search for domain knowledge from knowledge graphs. *arXiv preprint arXiv:2309.03118*, 2023.

Emilio Ferrara. Should chatgpt be biased? challenges and risks of bias in large language models. *arXiv preprint arXiv:2304.03738*, 2023.

Jessica Ficler and Yoav Goldberg. Controlling linguistic style aspects in neural language generation. *arXiv preprint arXiv:1707.02633*, 2017.

Isabel O Gallegos, Ryan A Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K Ahmed. Bias and fairness in large language models: A survey. *arXiv preprint arXiv:2309.00770*, 2023.

Leo Gao et al. The pile: An 800gb dataset of diverse text for language modeling, 2020.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. Creating training corpora for nlg micro-planning. In *55th annual meeting of the Association for Computational Linguistics (ACL)*, 2017.

Aparna Garimella, Rada Mihalcea, and Akhash Amarnath. Demographic-aware language model fine-tuning as a bias mitigation technique. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*, pages 311–319, 2022.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. *arXiv preprint arXiv:2009.11462*, 2020.

Sayan Ghosh, Mathieu Chollet, Eugene Laksana, Louis-Philippe Morency, and Stefan Scherer. Affect-lm: A neural language model for customizable affective text generation. *arXiv preprint arXiv:1704.06851*, 2017.

Michael Gira, Ruisu Zhang, and Kangwook Lee. Debiasing pre-trained language models via efficient fine-tuning. In *Proceedings of the Second Workshop on Language Technology for Equality, Diversity and Inclusion*, pages 59–69, 2022.

Seraphina Goldfarb-Tarrant, Rebecca Marchant, Ricardo Muñoz Sánchez, Mugdha Pandya, and Adam Lopez. Intrinsic bias metrics do not correlate with application bias. *arXiv preprint arXiv:2012.15859*, 2020.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*, 2023.

Anthony G Greenwald, Debbie E McGhee, and Jordan LK Schwartz. Measuring individual differences in implicit cognition: the implicit association test. *Journal of personality and social psychology*, 74(6):1464, 1998.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

Demi Guo, Alexander M Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. *arXiv preprint arXiv:2012.07463*, 2020.

Wei Guo and Aylin Caliskan. Detecting emergent intersectional biases: Contextualized word embeddings contain a distribution of human-like biases. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 122–133, 2021.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks, 2020.

Xudong Han, Timothy Baldwin, and Trevor Cohn. Towards equal opportunity fairness through adversarial learning. *arXiv preprint arXiv:2203.06317*, 2022.

Lukas Hauzenberger, Shahed Masoudian, Deepak Kumar, Markus Schedl, and Navid Rekabsaz. Modular and on-demand bias mitigation with attribute-removal subnetworks. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6192–6214, 2023.

Daniel Hershcovich, Stella Frank, Heather Lent, Miryam de Lhoneux, Mostafa Abdou, Stephanie Brandl, Emanuele Bugliarello, Laura Cabello Piqueras, Ilias Chalkidis, Ruixiang Cui, et al. Challenges and strategies in cross-cultural nlp. *arXiv preprint arXiv:2203.10020*, 2022.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification, 2018.

Huseyin A. Inan, Osman Ramadan, Lukas Wutschitz, Daniel Jones, Victor Rühle, James Withers, and Robert Sim. Training data leakage analysis in language models, 2021.

Nishtha Jain, Maja Popovic, Declan Groves, and Eva Vanmassenhove. Generating gender augmented data for nlp. *arXiv preprint arXiv:2107.05987*, 2021.

Przemyslaw Joniak and Akiko Aizawa. Gender biases and where to find them: Exploring gender bias in pre-trained transformer-based language models using movement pruning. *arXiv preprint arXiv:2207.02463*, 2022.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

Siwon Kim, Sangdoo Yun, Hwaran Lee, Martin Gubri, Sungroh Yoon, and Seong Joon Oh. Propile: Probing privacy leakage in large language models, 2023.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. Gedi: Generative discriminator guided sequence generation. *arXiv preprint arXiv:2009.06367*, 2020.

Julia Kreutzer, Isaac Caswell, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, et al. Quality at a glance: An audit of web-crawled multilingual datasets. *Transactions of the Association for Computational Linguistics*, 10:50–72, 2022.

Keita Kurita, Anna Belova, and Antonios Anastasopoulos. Towards robust toxic content classification. *arXiv preprint arXiv:1912.06872*, 2019.

Anne Lauscher, Tobias Lueken, and Goran Glavaš. Sustainable modular debiasing of language models. *arXiv preprint arXiv:2109.03646*, 2021.

Nayeon Lee, Wei Ping, Peng Xu, Mostofa Patwary, Pascale N Fung, Mohammad Shoeybi, and Bryan Catanzaro. Factuality enhanced language models for open-ended text generation. *Advances in Neural Information Processing Systems*, 35: 34586–34599, 2022.

Alyssa Lees, Vinh Q Tran, Yi Tay, Jeffrey Sorensen, Jai Gupta, Donald Metzler, and Lucy Vasserman. A new generation of perspective api: Efficient multilingual character-level transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3197–3207, 2022.

Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. Halueval: A large-scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6449–6464, 2023a.

Shaobo Li, Xiaoguang Li, Lifeng Shang, Zhenhua Dong, Chengjie Sun, Bingquan Liu, Zhenzhou Ji, Xin Jiang, and Qun Liu. How pre-trained language models capture factual knowledge? a causal-inspired analysis. *arXiv preprint arXiv:2203.16747*, 2022.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Lidong Bing, Shafiq Joty, and Soujanya Poria. Chain of knowledge: A framework for grounding large language models with structured knowledge bases. *arXiv preprint arXiv:2305.13269*, 2023b.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023c.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

Adam Liska, Tomas Kocisky, Elena Gribovskaya, Tayfun Terzi, Eren Sezener, Devang Agrawal, D'Autume Cyprien De Masson, Tim Scholtes, Manzil Zaheer, Susannah Young, et al. Streamingqa: A benchmark for adaptation to new knowledge over time in question answering models. In *International Conference on Machine Learning*, pages 13604–13622. PMLR, 2022.

Haochen Liu, Da Tang, Ji Yang, Xiangyu Zhao, Hui Liu, Jiliang Tang, and Youlong Cheng. Rating distribution calibration for selection bias mitigation in recommendations. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 2048–2057, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450390965. doi: 10.1145/3485447.3512078. URL https://doi.org/10.1145/3485447.3512078.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

Kaiji Lu, Piotr Mardziel, Fangjing Wu, Preetam Amancharla, and Anupam Datta. Gender bias in neural natural language processing. *Logic, Language, and Security: Essays Dedicated to Andre Scedrov on the Occasion of His 65th Birthday*, pages 189–202, 2020.

Ziyang Luo, Can Xu, Pu Zhao, Xiubo Geng, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Augmented large language models with parametric knowledge guiding. *arXiv preprint arXiv:2305.04757*, 2023.

Potsawee Manakul, Adian Liusie, and Mark JF Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*, 2023.

Justus Mattern, Zhijing Jin, Mrinmaya Sachan, Rada Mihalcea, and Bernhard Schölkopf. Understanding stereotypes in language models: Towards robust measurement and zero-shot debiasing. *arXiv preprint arXiv:2212.10678*, 2022.

Rowan Hall Maudslay, Hila Gonen, Ryan Cotterell, and Simone Teufel. It's all in the name: Mitigating gender bias with name-based counterfactual data substitution. *arXiv preprint arXiv:1909.00871*, 2019.

Chandler May, Alex Wang, Shikha Bordia, Samuel R Bowman, and Rachel Rudinger. On measuring social biases in sentence encoders. *arXiv preprint arXiv:1903.10561*, 2019.

Nick McKenna, Tianyi Li, Liang Cheng, Mohammad Javad Hosseini, Mark Johnson, and Mark Steedman. Sources of hallucination by large language models on inference tasks. *arXiv preprint arXiv:2305.14552*, 2023.

Nicholas Meade, Elinor Poole-Dayan, and Siva Reddy. An empirical survey of the effectiveness of debiasing techniques for pre-trained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1878–1898, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.132. URL https://aclanthology.org/2022.acl-long.132.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. Augmented language models: a survey, 2023.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. *arXiv preprint arXiv:2305.14251*, 2023.

Niloofar Mireshghallah, Hyunwoo Kim, Xuhui Zhou, Yulia Tsvetkov, Maarten Sap, Reza Shokri, and Yejin Choi. Can llms keep a secret? testing privacy implications of language models via contextual integrity theory. *arXiv preprint arXiv:2310.17884*, 2023.

Niels Mündler, Jingxuan He, Slobodan Jenko, and Martin Vechev. Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation. *arXiv preprint arXiv:2305.15852*, 2023.

Moin Nadeem, Anna Bethke, and Siva Reddy. Stereoset: Measuring stereotypical bias in pretrained language models. *arXiv preprint arXiv:2004.09456*, 2020.

Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R Bowman. Crows-pairs: A challenge dataset for measuring social biases in masked language models. *arXiv preprint arXiv:2010.00133*, 2020.

Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from (production) language models, 2023.

Roberto Navigli and Simone Paolo Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial intelligence*, 193:217–250, 2012.

Roberto Navigli, Simone Conia, and Björn Ross. Biases in large language models: Origins, inventory and discussion. *ACM Journal of Data and Information Quality*, 2023.

Debora Nozza, Federico Bianchi, Dirk Hovy, et al. Honest: Measuring hurtful sentence completion in language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2021.

Ali Omrani, Alireza Salkhordeh Ziabari, Charles Yu, Preni Golazizian, Brendan Kennedy, Mohammad Atari, Heng Ji, and Morteza Dehghani. Social-group-agnostic bias mitigation via the stereotype content model. In *Proc. The 61st Annual Meeting of the Association for Computational Linguistics (ACL2023)*, 2023.

Hadas Orgad and Yonatan Belinkov. Blind: Bias removal with no demographics. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8801–8821, 2023.

SunYoung Park, Kyuri Choi, Haeun Yu, and Youngjoong Ko. Never too late to learn: Regularizing gender bias in coreference resolution. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 15–23, 2023.

John Pavlopoulos, Jeffrey Sorensen, Lucas Dixon, Nithum Thain, and Ion Androutsopoulos. Toxicity detection: Does context really matter? *arXiv preprint arXiv:2006.00998*, 2020.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*, 2023.

Ethan Perez, Sam Ringer, Kamilė Lukošiūtė, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, et al. Discovering language model behaviors with model-written evaluations. *arXiv preprint arXiv:2212.09251*, 2022.

Yusu Qian, Urwa Muaz, Ben Zhang, and Jae Won Hyun. Reducing gender bias in word-level language models with a gender-equalizing loss function. *arXiv preprint arXiv:1905.12801*, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016.

Leonardo Ranaldi, Elena Sofia Ruzzetti, Davide Venditti, Dario Onorati, and Fabio Massimo Zanzotto. A trip towards fairness: Bias and de-biasing in large language models. *arXiv preprint arXiv:2305.13862*, 2023.

Ruiyang Ren, Yuhao Wang, Yingqi Qu, Wayne Xin Zhao, Jing Liu, Hao Tian, Hua Wu, Ji-Rong Wen, and Haifeng Wang. Investigating the factual knowledge boundary of large language models with retrieval augmentation, 2023.

Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

Yash Savani, Colin White, and Naveen Sundar Govindarajulu. Intra-processing methods for debiasing neural networks, 2020.

Timo Schick, Sahana Udupa, and Hinrich Schütze. Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in nlp, 2021.

John Schulman. Reinforcement learning from human feedback: progress and challenges. In *Berkley Electrical Engineering and Computer Sciences. URL: https://eecs. berkeley. edu/research/colloquium/230419 [accessed 2023-11-15]*, 2023.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*, 2023.

Eric Michael Smith, Melissa Hall, Melanie Kambadur, Eleonora Presani, and Adina Williams. "i'm sorry to hear that": Finding new biases in language models with a holistic descriptor dataset. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9180–9211, 2022.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/D13-1170`.

Irene Solaiman and Christy Dennison. Process for adapting language models to society (palms) with values-targeted datasets. *Advances in Neural Information Processing Systems*, 34:5861–5873, 2021.

Robin Staab, Mark Vero, Mislav Balunović, and Martin Vechev. Beyond memorization: Violating privacy via inference with large language models, 2023.

Tianxiang Sun, Xiaotian Zhang, Zhengfu He, Peng Li, Qinyuan Cheng, Hang Yan, Xiangyang Liu, Yunfan Shao, Qiong Tang, Xingjian Zhao, et al. Moss: Training conversational language models from synthetic data. *arXiv preprint arXiv:2307.15020*, 7, 2023.

Tony Sun, Kellie Webster, Apu Shah, William Yang Wang, and Melvin Johnson. They, them, theirs: Rewriting with gender-neutral english, 2021.

Himanshu Thakur, Atishay Jain, Praneetha Vaddamanu, Paul Pu Liang, and Louis-Philippe Morency. Language models get a gender makeover: Mitigating gender bias with few-shot data interventions. *arXiv preprint arXiv:2306.04597*, 2023.

Ewoenam Kwaku Tokpo and Toon Calders. Text style transfer for bias mitigation using masked language modeling. *arXiv preprint arXiv:2201.08643*, 2022.

Hugo Touvron et al. Llama 2: Open foundation and fine-tuned chat models, 2023.

Prasetya Ajie Utama, Nafise Sadat Moosavi, and Iryna Gurevych. Towards debiasing nlu models from unknown biases. *arXiv preprint arXiv:2009.12303*, 2020.

Pranav Narayanan Venkit, Sanjana Gautam, Ruchi Panchanadikar, Shomir Wilson, et al. Nationality bias in text generation. *arXiv preprint arXiv:2302.02463*, 2023.

Eric Wallace, Tony Z Zhao, Shi Feng, and Sameer Singh. Concealed data poisoning attacks on nlp models. *arXiv preprint arXiv:2010.12563*, 2020.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2019.

Xinyi Wang, Sebastian Ruder, and Graham Neubig. Expanding pretrained models to thousands more languages via lexicon-based adaptation. *arXiv preprint arXiv:2203.09435*, 2022.

Kellie Webster, Xuezhi Wang, Ian Tenney, Alex Beutel, Emily Pitler, Ellie Pavlick, Jilin Chen, Ed Chi, and Slav Petrov. Measuring and reducing gendered correlations in pre-trained models. *arXiv preprint arXiv:2010.06032*, 2020.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Wikipedia Contributors. Who writes wikipedia? https://en.wikipedia.org/wiki/Wikipedia:Who_writes_Wikipedia%3F, 2023. Accessed: 2023-04-01.

Weiqi Wu, Chengyue Jiang, Yong Jiang, Pengjun Xie, and Kewei Tu. Do plms know and understand ontological knowledge? *arXiv preprint arXiv:2309.05936*, 2023.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. *arXiv preprint arXiv:2306.13063*, 2023.

Ke Yang, Charles Yu, Yi R Fung, Manling Li, and Heng Ji. Adept: A debiasing prompt framework. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10780–10788, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. Do large language models know what they don't know? *arXiv preprint arXiv:2305.18153*, 2023.

Charles Yu, Sullam Jeoung, Anish Kasi, Pengfei Yu, and Heng Ji. Unlearning bias in language models by partitioning gradients. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6032–6048, 2023.

Sina Zarrieß, Henrik Voigt, and Simeon Schüz. Decoding methods in neural language generation: a survey. *Information*, 12(9):355, 2021.

Abdelrahman Zayed, Goncalo Mordido, Samira Shabanian, and Sarath Chandar. Should we attend more or less? modulating attention for fairness. *arXiv preprint arXiv:2305.13088*, 2023.

Yuheng Zha, Yichi Yang, Ruichen Li, and Zhiting Hu. Alignscore: Evaluating factual consistency with a unified alignment function. *arXiv preprint arXiv:2305.16739*, 2023.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. Siren's song in the ai ocean: A survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*, 2023.

Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. *arXiv preprint arXiv:2305.03268*, 2023.

Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. Dense text retrieval based on pretrained language models: A survey. *arXiv preprint arXiv:2211.14876*, 2022.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*, 2023.

Chapter 7
Retrieval-Augmented Generation

Abstract Retrieval-augmented generation (RAG) is a prominent application of conversational LLMs. RAG systems accept a user query and return a response similar to chatbots but source the factual details of their response from static knowledge bases, including documents, structured data tables, and more. In RAG, a small language model is used to embed the user query and compare it against a similarly embedded document corpus to find semantically similar, and thus contextually relevant, text segments. The original query and the retrieved documents are then passed to an instruction-tuned chatbot, which uses the documents to answer the query. In this chapter, we discuss the basic underpinnings of RAG and describe the details that an engineer must consider when building a RAG system. We then do deep dives on many modular enhancements that can be incorporated into a RAG workflow to expand capabilities and safeguard against weaknesses. Next, we discuss several test metrics commonly used for evaluating RAG performance, probing both the accuracy of dense retrieval and the success of chatbots in answering queries. Finally, we present a tutorial for building, enhancing, and evaluating a RAG system using the LlamaIndex codebase.

7.1 Introduction

One of the most appealing applications of auto-generative LLMs is as a general knowledge base that can be queried with natural language and replies with factual responses. As we discussed in Sect. 3.2.3, LLMs memorize large volumes of information within their billions of parameters, and with a well-written prompt (and a little bit of luck), users can sometimes elicit accurate responses. However, the accuracy of a given response may be compromised by inaccurate, incomplete, or absent information in the training corpus or by LLM hallucinations (see Sect. 6.1). As such, the real-world actionability of such information is strongly dependent on the toler-

U. Kamath et al., *Large Language Models: A Deep Dive*,
https://doi.org/10.1007/978-3-031-65647-7_7

ance level for inaccuracies. Factual errors in LLM responses may not be tolerable in a given application, such as an educational chatbot, medical diagnoses, or automated customer service agents.

Retrieval-Augmented Generation (RAG) has been developed to mitigate these problems of inaccurate or hallucinatory recall. At the most basic level, the RAG approach uses LLMs to create embedding representations of the text within a database of reliable information, rapidly searches for and locates passages responsive to a given query, and return the information in a form useful to the user. In essense, a RAG system is a QA chatbot that sources information from a fixed database instead of relying on pre-training to memorize factual details. This makes it both more reliable in its returned information and extensible to documents that were not part of the LLM pre-training dataset.

RAG was originally introduced in Lewis et al. (2020). However, since the popularization of ChatGPT and similar high-performing chatbots and the realization of their superior ability to reason in-context, research and innovation in RAG techniques have exploded as researchers and developers have worked to solve and optimize the various functional components of the framework. In this chapter, we summarize the essential points of RAG, discuss a number of improvements developed in the recent literature for extending functionality and improving the performance of RAG systems, and overview approaches for evaluating the performance of a RAG system. We will close with a tutorial where we build a RAG system using the popular LlamaIndex package (Liu, 2022) and experiment with a few augmentations.

7.2 Basics of RAG

At its core, a basic RAG system executes the four steps represented graphically in Fig. 7.1:

1. **Indexing:** A series of documents are chunked into text segments, and each segment is transformed into a text embedding with a chosen LLM. These embeddings are placed in a vector index where they can be rapidly compared for semantic similarity against additional vectors.
2. **Querying:** A user enters a query that is answerable based on the content of the documents, and this query is embedding using the same embedding model as was used to build the vector index of documents.
3. **Retrieval:** The transformed query is compared against each embedded segment in the vector index, typically using cosine distance, and the segments are ranked by their similarity to the query. The few top-scoring segments are then extracted in their original text representation. Ideally, the most similar chunks will contain information pertinent to the query.
4. **Generation:** These top segments are packaged in a prompting template as context, along with the original query, and the template is passed to an

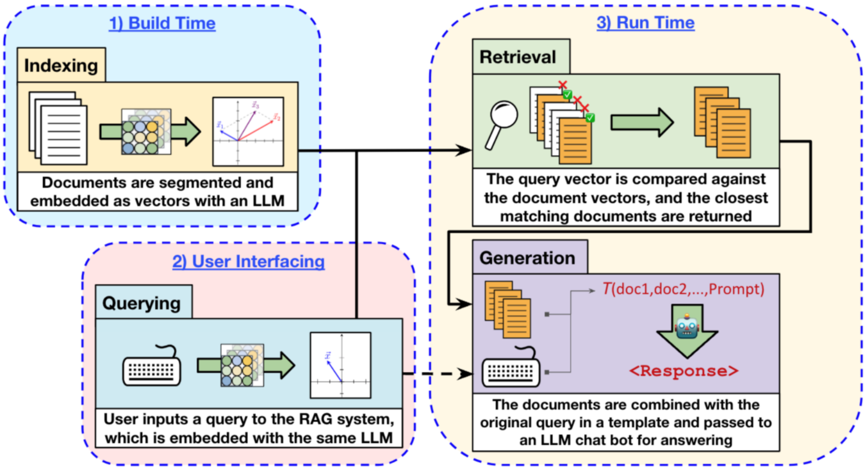

1) Build Time
Indexing
Documents are segmented and embedded as vectors with an LLM
2) User Interfacing
Querying
User inputs a query to the RAG system, which is embedded with the same LLM
3) Run Time
Retrieval
The query vector is compared against the document vectors, and the closest matching documents are returned
Generation
T(doc1,doc2,...,Prompt)
<Response>
The documents are combined with the original query in a template and passed to an LLM chat bot for answering

Fig. 7.1: The basic conceptual workflow for a RAG system, including initial document vectorization and indexing, user querying, retrieval, generation, and output. The system locates useful documents within its corpus and passes these documents along with the original query to the generator to create a knowledge-based response to the query.

LLM-based QA agent. The agent then answers the question based on the retrieved context, and the user is given the output.

Fig. 7.2 illustrates a concrete example of the basic RAG cycle. We want a response to the following question:

```
Who owns the content created by OpenAI programs?
```

If we ask ChatGPT this question, it responds that:

```
OpenAI typically retains ownership of the content...
```

However, this is not true. Instead, let us take the OpenAI terms of service, segment the documents, and create a vector index. When we then query this vector index with the embedded question, we find two chunks specifically detailing ownership rights over outputs from OpenAI services. These documents are placed into a fixed template that includes them as context prior to asking the question. When this templatized version is passed to ChatGPT, it now responds:

```
As per the provided context information, users own any output
    they rightfully receieve from OpenAI services.
```

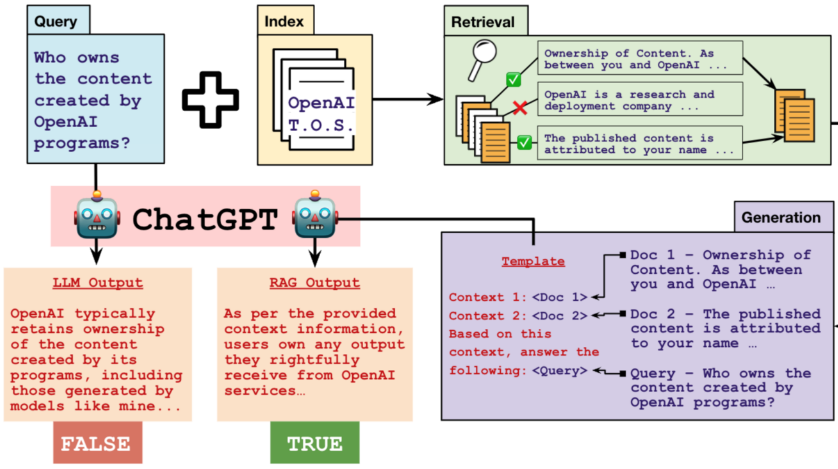

Query
Who owns the content created by OpenAI programs?
Index
OpenAI T.O.S.
Retrieval
Ownership of Content. As between you and OpenAI ...
OpenAI is a research and deployment company ...
The published content is attributed to your name ...
ChatGPT
Generation
Template
Context 1: <Doc 1>
Context 2: <Doc 2>
Based on this context, answer the following: <Query>
Doc 1 - Ownership of Content. As between you and OpenAI ...
Doc 2 - The published content is attributed to your name ...
Query - Who owns the content created by OpenAI programs?
LLM Output
OpenAI typically retains ownership of the content created by its programs, including those generated by models like mine...
RAG Output
As per the provided context information, users own any output they rightfully receive from OpenAI services...
FALSE
TRUE

Fig. 7.2: Practical illustration of the RAG workflow, using a question about the ownership of OpenAI output as an example. If we ask ChatGPT the question, we get the wrong answer. Instead, a RAG call with a vector index of the OpenAI terms of service gives the correct answer.

This is the correct answer, which ChatGPT has now been able to report due to our RAG system. (See Sect. 7.6 for full tutorial)

We accomplish a few things by constructing a system that bases its response on a source of information external to the training weights. First, we can use LLM-style semantic reasoning on data that was never part of the original training data. This is critical because LLMs always have cutoff dates for their training set, which prevents them from answering questions about recent events. This is shown in the above example, where the generator bot cannot answer the question about OpenAI's terms of service, which had been updated since the ChatGPT training epoch. Second, because we are passing the relevant context directly to the generator and asking it to answer the query based only on this information, we can increase the accuracy and precision of the response compared to simply trusting our LLMs as knowledge repositories. Many SOTA LLM chatbots have fairly opaque training datasets, and it is not always clear what information they know or how reliably it can be accessed, and they are prone to hallucinate and confidently report things they do not know. In-context reasoning provides more reliable answers than does relying on the correct expression of pre-trained information.

! Practical Tips

As promising as this sounds, several challenges make RAG systems difficult to perfect. Many parameters and approaches control each of the steps listed above, and the model will not work optimally without prudent choices of these configurations. Crucially, you must ensure that the vector search correctly identifies relevant chunks of text and that you know how to query the generator to extract that information appropriately. Without these optimizations, RAG systems may be as likely to hallucinate as normal LLM calls; to this end, RAG systems must also be taught to admit when they do not know the answer. The cost of failure can be high, as seen in a recent episode in which Air Canada was forced to honor a nonexistent (i.e., hallucinated) policy described to a customer by its AI-based chatbot[1].

7.3 Optimizing RAG

The basic structure of a RAG application is conceptually straightforward, but many details must be worked out. Each of the four steps described in Sect. 7.2 contains a number of preprocessing steps, architectural choices, and hyperparameter settings that must be defined. Prudent choices of these enhancements are essential for optimizing the performance of a RAG system. In this section, we provide an overview of the different procedures implicit in the basic RAG steps and discuss the parameters that should be considered for each step when constructing a RAG system. A tabular summary of these details is given in Table 7.1. First, while building out the vector index, the following considerations apply:

- **Text preprocessing** – This optimization entails the conversion of raw documents into an ideal format. Word documents, HTML pages, PDFs, epubs, and other formats should be converted to plain text. Embedded tables and markdown format should be converted into LLM-friendly input (see below). The text may also be cleaned of extraneous information and normalized to ensure accurate semantic matching. Note that any document normalization must also be performed on the input query at runtime.
- **Text chunking** – This step involves breaking the documents into smaller chunks of similar size for embedding. There are trade-offs to consider when determining the size of each chunk. Smaller chunks allow for more granular information in each vector but may also be too small to capture the relevant context. Smaller chunk sizes also produce more total chunks, thus resulting in slower searches. Chunking can be performed at a fixed character or word length, by sentences, paragraphs, sections, or documents, or by a token length that optimizes the rate of LLM encoding.
- **Metadata** – Chunks can also be augmented with useful metadata, including document titles, authors, subjects, dates, or any other categories that differentiate one document from another. These facts may be used to filter for relevant vectors before the search, or the metadata may be included within each text chunk to add extra context to the vector embeddings.

[1] https://arstechnica.com/tech-policy/2024/02/air-canada-must-honor-refund-policy-invented-by-airlines-chatbot/

- **Embedding model** – The choice of the model determines how effectively the RAG system can retrieve chunks responsive to queries. This is a semantic textual similarity NLP task, so models should be chosen appropriately[2]. Larger models will typically produce richer embeddings, while the number of parameters, vector dimensions, and embedding latency determine the expense of computation. The model's context window size is also relevant as a cap on chunk length. While most of the computational overhead occurs when embedding the documents, this choice also determines what embedding model is used on the query at runtime.
- **Index storage** – Many options exist, with relevant trade-offs including search speed, scalability, static databases vs. expandable databases, open source vs. proprietary, and centralized vs. distributed. Superlinked[3] has created and maintained a useful table of vector databases, comparing features and performance.

Each of these steps defines how the documents are handled and stored. Next, we look at how these databases are queried and how the retrieved documents are used for answer generation:

- **Retrieval function** – Similarity between prompt and text chunks is generally determined by cosine distance, but the quantity k of the top documents to return is tunable. A small k provides s shorter context for the generation step, which can improve LLM comprehension but may also leave out relevant information contained in documents with slightly lower scores. A large k passes more information to the generation step but increases the risk of irrelevant information diluting the desired signal.
- **Generation architecture** – Architectural choices for generation include which LLM to use, what prompt template to use for combining query and context, and what system instructions to pass before the query/context portion. Optimal LLMs for the generation step are large, instruction-tuned chat-bots such as ChatGPT, Claude, or Llama-2. Cost is a significant consideration here, with a trade-off against quality – at the time of writing, GPT-4 API calls cost roughly 50 times GPT-3.5-turbo API calls per token, but provide superior performance in generative tasks.
- **Context formatting** – how to combine potentially disparate top-k documents into a coherent context for the chatbot. Choices include providing all documents in a list, summarizing each with the generator LLM to better fit a context window, or using the generator LLM to consolidate the chunks into a single paragraph of known information.

This overview is not exhaustive but provides a strong starting point for the baseline requirements to consider when creating a RAG system. In the next section, we will detail a number of enhancements that can be added to this picture to increase functionality, improve performance, and broaden the scope

[2] HuggingFace maintains a leader board benchmarking STS performance against the MTEB dataset (Muennighoff et al., 2023), useful for RAG applications – `https://huggingface.co/spaces/mteb/leaderboard`

[3] `https://superlinked.com/vector-db-comparison`

7.4 Enhancing RAG

In late-2022, ChatGPT and its competitors revolutionized overnight the quality of outputs from the generation step of RAG applications. Perhaps more importantly, their impressive human-like responses to factual questions created broad demand in the market for chatbots that can work with information not included in their training data. These developments opened the door for a flourishing of new RAG applications, and the design of new modules and procedures that can be integrated into the RAG workflow. Gao et al. (2024) have coined the term "modular RAG" to describe systems built with these new innovations. In this section, we discuss a number of these improvements, listing them chronologically in the RAG workflow, from indexing to querying to retrieval and, finally, to generation. We focus particularly on the specific performance issues they are meant to address. Cartoon illustrations of each optimization and enhancement we discuss are shown at their approximate locations in the RAG pipelines in Figs. 7.3-7.6.

7.4.1 Data Sources and Embeddings

The most straightforward improvements to the data indexing stage are standard data sanitation practices, such as input text normalization, stripping extraneous markings like HTML tags, and optimizing segmentation size. However, there are more complex enhancements that can boost model performance and breadth of knowledge. Here, we briefly detail approaches for including structured data tables in a RAG indexing system, and discuss the advantages of fine-tuning the indexing embedding model.

7.4.1.1 Use of Structured Data

Many sources of information that could benefit from RAG-style querying come in formats that are ill-suited for transformation into plain text. These include data tables in documents, SQL databases, knowledge bases, and websites. Data structures are a vital source of factual, numerical, and comparative information that RAG applications must be able to interpret correctly. Here, we review existing approaches for incorporating this information.

One tactic, explored in several works (Hu et al., 2023; Wang et al., 2023d) is to integrate table querying into the retrieval portion of a RAG application. In this approach, a set of documents can be enhanced with, for example, a SQL table containing additional relevant information. Then, a RAG system is equipped with a router (see Sect. 7.4.2.4 below) that determines whether a specific user query would benefit from information in the table. If so, it passes the user query to an LLM trained on SQL code, which generated a fit-to-purpose SQL call. The table is then queried with

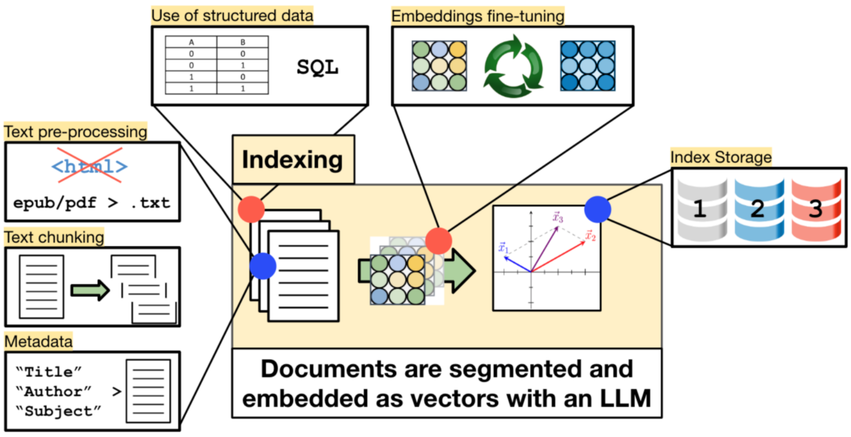


Fig. 7.3: Illustration of the different enhancements discussed for RAG indexing. Preprocessing, chunking, and metadata (Sect. 7.3) operate on the corpus prior to embedding, and can be enhanced with structured data (Sect. 7.4.1.1). Fine-tuning of the embeddings (Sect. 7.4.1.2) and a prudent choice of index storage (Sect. 7.3) can optimize the retrieval accuracy and throughput.

this output. The returned information is then passed along with the query (and any other retrieved documents) to the generator to produce a final response.

As a concrete example, Hu et al. (2023) give the following input/output pair to demonstrate the capabilities of their *ChatDB* system:

```
Question: What was the total revenue for January 2023?

LLM-generated Query:
SELECT SUM(total_price) AS total_revenue
FROM sales
WHERE sale_date >= '2023-01-01' AND sale_date < '2023-02-01';

Database response:
+----------------+
| total_revenue |
+----------------+
|     707.0      |
+----------------+
```

The system converts a plain English request into a precise SQL query designed to return the relevant information, which can serve as the basis for a generated answer.

! Practical Tips

Not all tables come in convenient searchable formats. In particular, when ingesting technical PDFs or similar documents, a RAG system will frequently come across tables containing valuable information. However, it is not obvious how to convert these tables into a well-suited representation for RAG. Little value can be achieved without a proper structure that retains relationships between table cells and their labels. In response, a number of solutions have been proposed to render PDF tables in a more retriever-friendly format. LlamaParse[4] is a recent development that uses a proprietary algorithm to parse a diverse array of table shapes to a markdown representation that retains the relationship between table quantities and their row/column labels. These can be integrated with iterative retrieval methods optimized for markdown, which can faithfully extract data relations for generation.

7.4.1.2 Embeddings Fine-tuning

The retrieval accuracy depends on how well the embedding model expresses the critical features of the RAG documents and, thus, how well they can be retrieved. Several open-source embedding models that excel at semantic textual similarity tasks, such as the BGE (Xiao et al., 2023) and VoyageAI (Wang et al., 2023a) series, have been released in recent years; however, given the generality of their training corpora, performance may degrade for subjects with specialized terminology and concepts. This issue can be addressed by fine-tuning the embeddings with domain-specific examples.

A popular approach, implemented in LlamaIndex (Liu, 2022), constructs training examples from the RAG documents themselves. Text chunks from a holdout set are passed to GPT-4, which instructs the creation of individual questions answered by the documents. The embeddings are then fine-tuned so that the retriever selects the correct source document for each generated question. This approach introduces the embedding model to specialized terminology and better adapts the model to bridge the semantic gap between queries and the style of chunking selected for the RAG model. Once the model has been tuned, the documents can be re-embedded, and the RAG application can be constructed. This approach has been shown to improve retrieval accuracy by 5-10%[5] compared to using base embeddings while improving performance on specific niche topics.

7.4.2 Querying

The central challenge in RAG systems is finding the relevant documents based on a human-written query. However, the wide variation in diction between users and the basic discrepancy between the grammatical and informational content of queries and the documents used to answer them complicate mat-

[4] https://www.llamaindex.ai/blog/introducing-llamacloud-and-llamaparse-af8cedf9006b

[5] https://blog.llamaindex.ai/fine-tuning-embeddings-for-rag-with-synthetic-data-e534409a3971

ters. Querying augmentation generally focuses on transforming human-written queries into a form more likely to match the proper chunks in the vector index. Here, we describe a few approaches for parsing, transforming, and extending to suit the user's needs better.

7.4.2.1 Query Rewriting

As we have discussed elsewhere in this book, LLM responses can be susceptible to the details of the input prompt. Minor variations in diction, grammar, etc., can elicit very different outcomes when passed to an autogenerative model. This is not a desirable outcome for RAG systems, where we want faithful responses to user queries even when they are suboptimally composed. One way to improve these odds is automated *query rewriting*, which transforms human-written queries into a prompt more likely to elicit the desired search results.

We generally do not know *a priori* the most effective form of prompting, so the optimal strategy for query rewriting is to train a rewriter to perform the transformation using reinforcement learning based on RAG pipeline outcomes. In Ma et al.

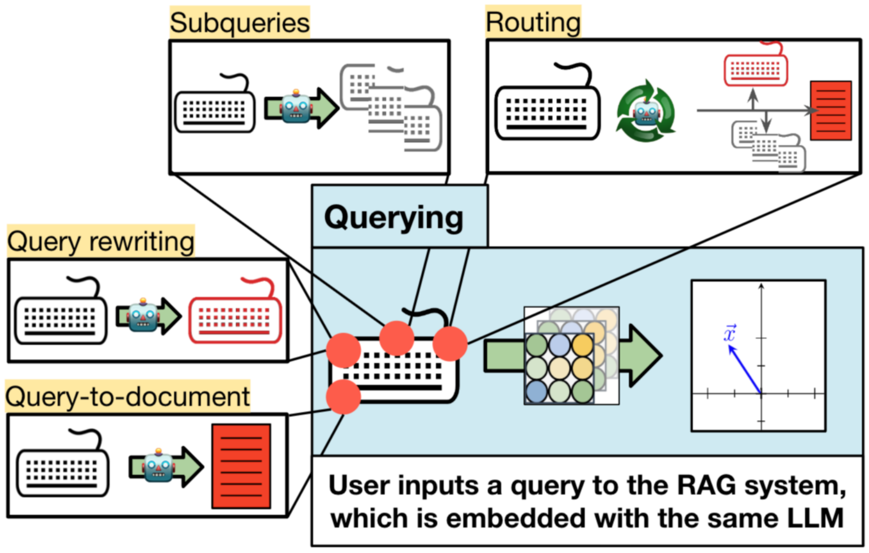

Subqueries
Routing
Querying
Query rewriting
Query-to-document
User inputs a query to the RAG system, which is embedded with the same LLM

Fig. 7.4: An illustration of the different enhancements discussed for RAG querying. Query rewriting (Sect. 7.4.2.1) and query-to-document expansion (Sect. 7.4.2.2) alter the user prompt using tuned LLMs to increase the likelihood of accurate document retrieval. Subquery generation (Sect. 7.4.2.3 uses an LLM to split complex prompts into component questions that can be queried in the RAG database more easily. Routing (Sect. 7.4.2.4) determines which of these enhancements to apply based on the content of the query.

(2023a), the authors defined a query rewriter using the *T5* model (Raffel et al., 2020), and tuned it using various QA training sets and a reward model based on the accuracy of the generator output. The result is a module sitting between the querying and retrieval stage, which converts the human-written query into an optimized form before embedding. They show improved performance for a trainable rewriter over a static rewriter (i.e., one that was defined but not fine-tuned) and no rewriter at all, demonstrating the value of this approach.

An alternative method for query rewriting was proposed by Raudashcl[6], who developed *RAG-Fusion*. In this approach, an initial query on a database is sent to ChatGPT, which then rewrites the prompt into several variants. The database is queried with each individual variant, and the output documents for each are merged into a single ranking through reciprocal rank fusion (RRF). In RRF, each document returned by a given search query is assigned a score given by:

$$\mathsf{Score}_{\mathsf{RRF}} = 1/(\mathsf{r}+60), \tag{7.1}$$

where r is that document's place in the search rankings. The scored results are then merged into a single list by summing the scores of any documents present in multiple searches. Thus, the highest scoring document in the RRF merge tends to be highly ranked in multiple search variants. This process is intended to smooth out search variation resulting from word choice and return documents that more robustly address the query in multiple formulations.

7.4.2.2 Query-to-Document Expansion

Basic RAG uses an embedded query to scan a series of vectorized text chunks for the most cosine-similar results in the hope that they contain the specific information that can address the query. One confounding issue in this approach is that, typically, queries are grammatically dissimilar from segments of the chunked documentation. The hope is that if the chunk's subject matter is similar enough to the content of the query, it will produce a good match, but the disparate textual structure can degrade the performance.

Query-to-document expansion seeks to address this issue. In this approach, the user query is passed to an autogenerative model, and the model is asked to create a hypothetical chunk of text within which the answer to the query is found. This chunk is then vectorized with the embedding model and used to search the vector index for semantic similarity. This process is amusingly called generation-augmented retrieval, or GAR (Mao et al., 2021). These generated text chunks frequently contain misinformation as the LLM hallucinates the answer to the query. Nevertheless, it creates a block of text on the queried topic that should be closer in format to the documents we are searching. This generative model can be fine-tuned so that its output more closely resembles the RAG document chunks, or it can use few-shot in-context learning by packaging the query with sample chunks to pick up the salient

[6] https://towardsdatascience.com/forget-rag-the-future-is-rag-fusion-1147298d8ad1

properties of the target documents. Variants of this method have been proposed by Gao et al. (2023) as *HyDE* (Hypothetical Document Encoding) , and by Wang et al. (2023b) as *query2doc*. The latter found up to 15% improvement in performance on various dense retrieval tasks when applying their method.

7.4.2.3 Subquery Generation

One weakness of basic RAG is its restriction to semantic similarity matching on a single query. This approach is frequently insufficient for locating all the necessary information to answer more complex queries that require synthesizing multiple pieces of information that could be located in different portions of the RAG corpus. For example, consider a query on a series of reports on the United States Consumer Price Index (CPI), asking about the key drivers of inflation in March from 2020-2023. These values are collectively present in the documents but not in a single chunk or its surrounding context. The values must be extracted from each month's CPI reports. The semantic matching capabilities of a basic RAG system will not be sufficient to retrieve all the necessary information and successfully synthesize it in the generation stage.

Subquery generation was designed to address this shortcoming. This approach uses a sophisticated chatbot (e.g., ChatGPT) to break the original query into a series of prompts that each target a single piece of needed information. Each of these queries follows the retrieval > generation pipeline, and the responses to each are aggregated as context for the original query, which is then passed to a final generation stage. With this paradigm, we can imagine how a subquery engine would ingest and process the CPI request given above:

```
Input:
What were the key drivers of inflation in the month of March
from 2020-2023?

Subqueries:
1) What were the key drivers of inflation in March 2020?
2) What were the key drivers of inflation in March 2021?
3) What were the key drivers of inflation in March 2022?
4) What were the key drivers of inflation in March 2023?
```

These subqueries will be much more effective at targeting the needed information from the CPI documents and should provide a sampling of the most important drivers of inflation in each of the four months. These four responses can then be synthesized as context for the original query.

7.4.2.4 Routing

We have detailed several pathways that an RAG system might traverse when going from a user query to a retrieval action, including query rewriting, subquery gener-