These notes discuss ways of incorporating human-in-the-loop feedback in NLP. To start, we explore why we want human feedback in the first place; then, we investigate what kinds of human feedback are usable; further, we ask how we can incorporate the feedback; finally, we discuss when to use them and the considerations around those decisions. Throughout this discussion, we will emphasize the importance of granularity as a lens for comparing different approaches to incorporating human feedback.

1 Why? Reasons we need human feedback

There is a fundamental misalignment between fine-tuning objectives (e.g. next-token loss) and what we care about (e.g. generating high-quality text).

For example, next-token-prediction assigns equal weight to important (e.g., factual) and unimportant (e.g., stylistic) tokens.

In general, we want to better align models with human values. This can include aspects like performance (corresponding to expectations of “model behaviors”), fairness (corresponding to societal values), explainability (corresponding to what we consider to be good rationales), and personal beliefs (corresponding to individual values).

The kind of feedback that you can provide depends on what the model kind of feedback can actually take.

The way you incorporate feedback can vary significantly in granularity, such as changing the dataset, loss function, or parameter space [Chen et al., 2022].

2 What? Types of human feedback and their applications

Although nowadays we hear a ton about reinforcement learning from human feedback (RLHF), human-in-the-loop techniques have a long and rich history behind them. We’ll explore these through a set of three examples, and then discuss more generally about ways to think about different types of feedback.

2.1 Example 1: Document Classification [Godbole et al., 2004]

For example, Godbole et al. [2004] explored broadly the kinds of different human feedback which could be used in document classification. As highlighted in Figure 1 [Godbole et al., 2004], for
their task, they identified that the kinds of feedback can usefully be described according to their **granularity**. Naturally, keep in mind that many of these examples are fairly dated but they provide a useful starting point for thinking about the kinds of feedback that are available and how you might break them down:

- **Document-level interaction:**
  - Suggest documents: this might be seen as a form of “active learning”, choosing the most informative documents for the user to label.
  - Suggest labels: this could reduce the cognitive load of the user, by suggesting labels that are likely to be correct.
  - Check label consistency: this could be used to detect errors in the labeling.

- **Word-level interaction:**
  - Suggest influential terms: the model could highlight terms that the user should pay attention to.
  - Accept/add/remove engineered features: the user could provide additional features to the model, if the space of possible features is very large and the user has some domain knowledge. The model could also suggest features to the user.

- **Model and data-level exploration:** providing information to the user about the dataset and making it easy to understand the data makes it easier for the user to make decisions about the model.

### 2.2 Example 2: Human-in-the-Loop Parsing [He et al., 2016]

Parsing is a difficult task, and asking individual people to solve a complex parsing task all in one go as part of a labeling process can be difficult and error-prone. [He et al., 2016] highlighted that this task could be broken down into many subproblems. They provide the example of “Pam ate the cake on the table that I baked last night.” While asking crowdworkers to write a complete syntax tree for this sentence may be infeasible, one can instead highlight the uncertain aspects of the parsing: an English speaker can easily determine that it is less likely that "I baked a table" than that "I baked cake."

This also makes it much easier to incorporate crowd feedback: instead of comparing a bunch of potentially very different syntax trees, you can instead ask them to answer a question to which there are few possible answers. Small addendum: note this may inspire challenges; while the authors did provide the original sentence to the crowdworkers, one can imagine a situation where a participant may only pay attention to the simple question (to reduce cognitive load); as a result, one could imagine edge case examples that require the full context like garden path sentences (e.g. “the old man the boat”) may result in incorrect solutions.
2.3 Example 3: Interactive topic modeling [Hu et al., 2014]

Interactive topic modeling [Hu et al., 2014] extends the extremely widely used tool of latent Dirichlet allocation (LDA) [Blei et al., 2003]. For context, in short, LDA is a generative topic model where each document is represented as a combination of topics that are themselves represented by a collection of words.

For interactive topic modeling, you start with a vanilla LDA with a symmetric Dirichlet prior. This just means that each topic is represented as a weighted bag of words that ignores any relationships between them. However, imagine you have an “animal” topic: naturally, the presence of “dog” increases the likelihood of “bark” more than “cat” does (and vice-versa for “meow”), but the symmetric Dirichlet prior with a “flat” structure cannot incorporate this.

You can instead intuitively imagine building subtopics within topics, and this is an easy way to think about the way that interactive topic modeling incorporates human feedback. Specifically, they allowed people to “merge” sets of similar subtopics or “split” nodes otherwise, providing a user interface for this annotation. They used this to build tree-based topic models, extending the model used in Andrzejewski et al. [2009]. Thus, they were able to incorporate human priors on topic structures, iteratively informed by the learned model.

2.4 Types of feedback

Broadly, consider that we ultimately want a way to transform human feedback into changes to our models. Chen et al. [2022] visualizes this in terms of a feedback loop between models and preferences, with model updates that incorporating knowledge from machine learning practitioners and preferences which are acquired from domain experts. One way to categorize the feedback is, again, at the granularity at which it is provided.

- Observation-level feedback (local): For example, we might infer preferences from human judgments on individual examples (e.g. radiologists provide gold annotations for medical images, by manually segmenting X-rays). One upside of observation-level feedback is its precision, making it more straightforward to incorporate it into model training. A downside of observation-level feedback is that inferring the underlying rules is ambiguous and more prone to learning spurious and uninterpretable features.

- Domain-level feedback (global): We might incorporate high-level domain knowledge like useful features in a dataset, or in the case of radiologists, particularly important regions of X-rays to pay attention to. The upsides and downsides here are basically flipped relative to observation-level feedback: this allows for clear and interpretable features, but it may be less straightforward to actually incorporate them into the model.

1Indeed, besides BERT [Devlin et al., 2018], Attention is All You Need [Vaswani et al., 2017], and Gradient-based learning applied to document recognition [LeCun et al., 1998], I am not aware of a more highly-cited NLP paper.
3 How? Types of model updates

When thinking about how to incorporate feedback, it is again useful to think in terms of granularity, this time relative to the model-relevant features. In this section, we will discuss dataset-level, loss-level, and parameter-level changes that can be made based on human feedback, and provide examples of global and local approaches for each. These are easiest to discuss in a supervised learning context (where we minimize an objective function on a dataset with some ground truth), where we may use the following equation to describe model parameter optimization:

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \sum_{(x, y) \in D} L(x, y; \theta)$$

(1)

Note also that dataset → loss → parameters can itself be seen as a way of grouping updates in order of increasing granularity. In addition, the steps in the natural language processing loop where each aspect is relevant are highlighted in subsection titles, among 1) raw data → 2) data labeling → 3) model selection → 4) model training → 5) evaluation/deployment [Wang et al., 2021].

3.1 Dataset-level [Steps 1, 2, 5]

We consider some global and local examples of dataset-level changes:

- Global: Generally, global changes involve systematically adding data points
  - Data augmentation: for example, if you have a biased dataset, you might counterfactually augment it to help make models trained on it less biased [Zmigrod et al., 2019].
  - Resampling: by sampling new combinations of data which may not be present in the original dataset, you can encourage generalization. Akyurek et al. [2020] showed that this improves generalization on morphology learning by mixing and matching examples in a dataset, e.g. to simulate unseen applications of rare tenses.
  - Weak supervision: For example, Snorkel [Ratner et al., 2020] uses imperfect/noisy supervision to train models by giving users high-level abstractions. The user can propose a "labeling function" they have in mind and then the model can incorporate it in order to build a more robust system. For example, if you are trying to build a model to identify the severity of medical symptoms, as a domain expert, you might propose that if a symptom has "pneumo," then that’s abnormal. You could then use this to relabel the data, and then retrain the model to handle examples that were either not covered by your labels or where labeling functions conflicted.

- Local:
  - Active learning: the model selects the best examples to label. For example, uncertainty sampling. You want to choose ones where the model is most uncertain. There are many ways to estimate "usefulness."
3.1.1 On the Internet

Dataset-level augmentation has received a ton of attention, especially because so many models are trained on data from the internet, which is (notoriously) not the cleanest or best-aligned data source. There has been prior work looking to create a version of the internet that is less toxic than the actual internet, e.g. highlighting “Hope” on the internet rather than hate [Chakravarthi, 2020]. Other work has explored the value of active learning via internet retrieval for efficient representation learning [Tong and Chang, 2001, Li et al, 2023].

3.1.2 Active learning vs weak supervision

One might ask, why use weak supervision versus active learning? In short, they serve different purposes and sometimes weak and active learning may even be used together. Data augmentation may result in many useless examples, so active learning can be employed to select the best ones. Moreover, test sets are often very large (both increasingly often and increasingly large): as a result, it can also be valuable to identify reasonable subsets that will result in scores well-correlated with the final results. Extrapolating from a carefully chosen subset can be seen as a way of using active learning in order to perform a kind of weak supervision.

3.2 Loss-level [Step 3]

Loss-level updates change the loss function, e.g., by adding or removing regularization terms.

- Global: For example, with unlikelihood training [Welleck et al, 2019], we specifically lower the likelihood of some undesired set of tokens by adding a penalty term corresponding to that set of tokens. For example, you may have a frequency or presence penalty, discouraging repetition.

- Local: Not all examples will be equally valuable. For example, if you have a subset of your data collected from crowdworkers or novices and a subset collected by experts, you may weigh the expert subset more highly. This can improve performance, especially in limited-data regimes [Xu et al, 2021].

3.3 Parameter-space-level [Steps 1, 2, 5]

Parameter-space-level updates are also known as model editing. With model editing, you can change the model parameters, e.g., by training the model or a subset of its parameters.

- Global: e.g., the Concept Bottleneck Model [Koh et al, 2020] is an approach that focuses on intermediate features for downstream predictions. In particular, it requires the model to predict a set of interpretable features, which are then used to make the ultimate prediction. One encouraging feature is that, when incorrect, these intermediate features can be corrected by experts. For example, if a model incorrectly identifies a bone spur in an x-ray, which is
then used for a downstream prediction, a doctor can directly update this intermediate feature to correct the model.

• Local: e.g., one can apply "language patches" to fix specific mistakes the model makes. This is done by encoding the language feedback into a model that determines whether a patch applies and, if so, what changes it makes [Murty et al., 2022]. One reason that this approach (and related techniques like knowledge editing [De Cao et al., 2021]) has received attention recently is that people would like to correct mistakes made by large language models without needing to perpetually train them or retrain them from scratch.

4 When? Different techniques for different situations

4.1 What are some forms of feedback?
Examples of feedback include: Labelling additional data points; editing data points; changing data weights; binary/scaled user feedback; natural language feedback; code language feedback; defining, adding, and removing feature spaces; directly changing the objective function; directly changing model parameters.

In general, most people generally prefer easier-to-provide feedback: Natural language feedback > labeling > model manipulation. However, experts may sometimes prefer the reverse because of the level of precision each offers.

4.2 When to use each technique
When choosing the appropriate technique for a given situation, there are various aspects to consider, many of which have no clear answers.

• We don’t want to cognitively overload people, so we can only ask for so much feedback and we should try to make it as easy as possible when we do.

• Humans make mistakes, so handling these mistakes, like in Snorkel [Ratner et al., 2020], and building robust models is important.

• It’s not always obvious who to collect data from. Who is sufficiently “expert” for a given task? To what extent can you overcome this by collecting multiple responses (sometimes, the distribution of answers itself is important, in which case you can’t just take the majority vote!)? How do you solicit responses from groups that may be disproportionately affected by a model but may be cautious about providing their information or sharing their experience?

• How do we visualize and understand changes in models? How should these visualizations/explanations differ for model developers relative to model users?

• Can your data collection strategy evolve over time to become better? How and when?
• How should you share what you’ve done? When is open-sourcing safe/appropriate? What responsibility do we have to red-team the models we release?

In general, these are also closely related to the question of when you want global or local feedback, both of which come with tradeoffs. Global feedback is more explicit, requiring you to specify what you want, but it is also more intrusive and may be harder to translate into changes. Local feedback requires you to infer preferences, which can be wrong but is often easier to translate into model changes. Ultimately, maintaining a human-centered mindset when developing or evaluating algorithms should allow one to build models which better reflect essential values.

References


