# REFERENCE

# Reexpress one

## Welcome! You've come to the right place for reliable data analysis!

**Reexpress one** is the most advanced on-device natural language data analysis platform ever created. It provides all of the data analysis tools you need to quickly and effectively analyze your language data... and all without writing a single line of analysis code. This reference document provides a high-level conceptual overview for interpreting and analyzing the data partitions and their associated probabilities; provides the specification for the input data format; and explains how to enable reranking of semantic searches.

If you're just getting started, we recommend first reading through the *Quick Start Guide*, which is available on our website **https://re.express/**.

**Introspection.**

**Updatability.**

**Uncertainty.**

Similarity ⧉ Distance ⧈ Magnitude ⧈

**Reexpress AI** for your data

# Document Data Format

Reexpress one uses a simple JSON lines format for the input data. You can click Help in Data->Add to download an example template to disk. In that view, you can also download Swift and Python code to get started with properly formatting JSON objects per line using the standard JSON parsers.

Each line of an imported JSON lines file (file extension: `.jsonl`) must be a well-formed JSON object with at least the following **required properties**:

| Property name | Data type | Requirement |
|---|---|---|
| `id` | String | ≤ 250 characters |
| | *Important*: The id defines the uniqueness of a document. Each document must have a distinct id. We recommend using a Universally Unique Identifier (UUID) as the id string. Re-uploading a document with the same id will delete any data previously associated with that document and will automatically transfer it to the datasplit chosen when re-uploading. | |
| `label` | Number | label ∈ {-99, -1, 0,...,C-1), where C is the total number of classes in the project (e.g., C=2 for binary classification) |

| | | |
|---|---|---|
| | The value -1 is a required placeholder for documents that lack labels.<br><br>The values {0,...,C-1} are the main labels for the task. The number of values available depends on the task specified at project creation and cannot be subsequently changed. For example, a binary classification task uses the values {0,1}, whereas a 3-class classification task uses the values {0,1,2}. String label names can be uploaded for display purposes: Go to Data->Labels.<br><br>At least two examples for each class must be uploaded to the Training and Calibration sets to begin training. ***However, we generally recommend having at least 1000 examples per class in both the Training and Calibration sets to get reliable uncertainty estimates.***<br><br>The value -99, the out-of-distribution (OOD) label, is less commonly used. The model does not directly predict this value, and documents with this label are effectively treated the same as unlabeled documents, but it can be useful in some settings to have a distinct label for analysis purposes. Documents with this label do not directly participate in setting the parameters of the model, but are taken into consideration when determining **Similarity to Training (q)**. Use this label for any documents that may be seen in practice at test-time but are effectively unrelated to the task. Including such a document in Training with the -99 value (as opposed to simply deleting the document) will ensure that any matches at test time to the unusual document will have a correspondingly low q value. (The model cannot predict -99, so q cannot exceed the rank of the match to the unusual document.) | |
| `document` | String | ≤ 5,000 characters |
| | Keep in mind that 5000 characters typically far exceeds the 512 token limit seen by the model. Keyword searches (as in Explore) do, however, have access to all characters saved to the project file. | |

The following are **optional properties** that can be added to each input JSON line:

| Property name | Data type | Requirement |
|---|---|---|
| `prompt` | String | ≤ 250 characters |
| | The default prompt (chosen at project creation) is only used if this optional prompt property is not present in the JSON. (As a result, if the prompt property is present and has an empty string value, the document will have an empty string as the prompt.) | |
| `info` | String | ≤ 250 characters |
| | As with the group property, this text is not seen by the model, but it can be searched via keywords (as in Explore) to subset the data. It can be edited directly in the document details navigator and can be used as a memo field. | |

| | | |
|---|---|---|
| `group` | String | ≤ 250 characters |
| | As with the info property, this text is not seen by the model, but it can be searched via keywords (as in Explore) to subset the data. It can be edited directly in the document details navigator and can be used as a memo field. | |
| `attributes` | Number Array | ≤ 32 values |
| | **Reexpression Attributes**<br><br>Up to 32 numbers, each separated by a comma. For example: 0.0,-0.1,0.2<br><br>*The output of another language model, or any other auxiliary information about a document, can be provided as input here.*<br><br>It is not a formal requirement, but typically each value should be between -2.0 and 2.0. The order of the attributes is taken into account. The value 0.0 will implicitly be used for any attribute not provided.<br><br>Technical note: Each attribute must be within the range of a single-precision floating-point number. | |

The JSON properties need not be in the above order. Each line must be separated by a standard newline character:

```
\n
```

Each individual file can have no more than 250,000 lines, and each individual file must be less than 2,000 MB.

(Each datasplit can have no more than 250,000 documents. A total of 15 datasplits can be present in a single project file.)

The above character counts are determined by a simple String count in Swift, as in the following Swift (version 5.9) code:

```
func countOfCharacters(inString inputString: String) -> Int {
    return inputString.count
}
```

It is important to properly escape any special characters in uploaded strings. Helper functions to get started with the standard JSON parsers in Swift and Python, demonstrating saving a properly formatted JSON object per line, are available for download in Data->Add->Help.

# Label Display Names Data Format

Optionally, you can provide a JSON lines formatted file to change the display names for the class labels. In this way, you can change the label names displayed in the interface from the standard integers to strings of your choosing.

You can click Help in Data->Labels to download an example template to disk.

Each line of the label JSON lines file (file extension: `.jsonl`) must be a well-formed JSON object with at least the following **required properties**:

| Property name | Data type | Requirement |
|---|---|---|
| `label` | Number | label ∈ {0,...,C-1), where C is the total number of classes in the project (e.g., C=2 for binary classification) |
| | Only display names for the main labels for the task can be modified. The out-of-distribution (OOD) label, -99, and the placeholder label for unlabeled documents, -1, cannot be modified. | |
| `name` | String | > 0 characters and ≤ 100 characters |
| | Not every available label needs to be present in the file, but if present, the label display name cannot be blank.<br><br>The label text is only used for display purposes and is not used by the model. Although the label can be up to 100 characters, it is generally recommended to keep the labels concise—less than about 50 characters—to avoid truncation in the visual displays. | |

The JSON properties need not be in the above order. Each line must be separated by a standard newline character:

```
\n
```

The above character counts are determined by a simple String count in Swift, as in the document data format described above.

# Uncertainty Guide

*In short, partition the data, and take the frequency.*

Reexpress one provides a new standard for reliable uncertainty quantification from language models. The algorithms are an extension of a line of work originally started at Harvard University.

The calculations are very complex to generate, but not to worry, Reexpress one does all of the heavy lifting for you. A key benefit is that, by design, interpreting the resulting probability is conceptually straightforward from the perspective of an end-user, and Reexpress one further allows you to quickly examine other documents assigned the same probability.

The basic premise is to split the data into partitions based on the predicted class, the Similarity to Training (q), the Distance to Training (d), and the Magnitude of the Output (f). Additional information about these signals and partitions are available within the application (including the ability to examine the values at a per-document level); here, we focus on a high-level conceptual overview.

The data partitions formed by the above signals establish the "reference class"[1] for assigning a probability to each prediction. For the types of document classification tasks and data scale typical to be used in Reexpress one, controlling for these partitions dominates in importance essentially all other choices in determining the probability, since by doing so, we can also reliably control for subtle distribution shifts in the data that could otherwise cause estimates to go unexpectedly off the rails. Once we have assigned the reference class, we can then simply take the frequency of points in the calibration set for a given reference class as the probability of a prediction. To account for data scarcity, we also make a couple minor adjustments described next.

Similar to common distribution smoothing techniques in natural language processing, which are inspired by the work of Good 1953[2], we apply add-1 smoothing to the estimate. That is, we add 1 to the denominator when calculating the proportion of correct predictions in a given reference class. As a general rule of thumb, to ensure a reasonable sample size, it is best for the size of the reference class to be greater than at least 100 (and ideally, greater than 1000), in which case, such smoothing becomes

---

[1] As of writing, Wikipedia has a short but informative overview of what is meant by a "reference class" in Statistics: https://en.wikipedia.org/wiki/Reference_class_problem

[2] Good, I. J. "The Population Frequencies of Species and the Estimation of Population Parameters." *Biometrika* 40, no. 3/4 (1953): 237–64. https://doi.org/10.2307/2333344.

largely irrelevant. However, when the sample size is small, such smoothing adds a degree of slack to the estimate in the direction of avoiding overconfidence. We further restrict the displayed probability to a minimum probability of 0.01 and a maximum probability of 0.99. This is not an inherent limitation of the method, but rather a decision to ensure that the initial versions of the application are not used for high-risk applications. Future versions of Reexpress will be enabled to handle finer resolutions.

Unique to Reexpress one, we also assign a second-order estimate of the reliability of the calibration process itself. At a high-level, data partitions that reflect a looser connection to the observed training and calibration data (e.g., low values of q and f, high values of d, and/or small partition sizes in calibration) are empirically more susceptible to distribution shifts across datasplits. When the calibrated probability for a given document is not in the Highest calibration reliability subset, we recommend making use of the visual comparison tools in Compose to examine the data for distribution shifts relative to the Calibration set, and to use the document-level dense search feature to examine the nearest documents in the Training set. Ideally, for a document not in the Highest calibration reliability subset, additional data is then collected to lift the document into the Highest calibration reliability subset. In some cases, this may also necessitate moving to a stronger model, as by ensembling with a third-party neural network by adding Reexpression attributes to the document.

# Reranking

Reexpress one has sophisticated semantic search capabilities, which includes the option to upweight particular lexical terms to modify a search intent. Given that the embeddings used in search are derived from the trained models against your data, these search capabilities can be very effective for exploring your data without additional reranking.

In some cases, it may be useful to structure your data and classification task in a multi-task manner in order to enable reranking of the results of a semantic search. The basic idea is to structure training examples to resemble the format of documents to be reranked (namely, the search text + the retrieved document, optionally with a prompt).

For example, suppose you have a standard binary classification task (e.g., predicting whether or not a customer's follow-up message reflects a positive interaction with your organization), and your organization has additionally constructed questions about those documents (e.g., as collected while examining the documents in Reexpress one[3]). The labels for these questions+document pairs would have the semantic meaning of the set {`Not Relevant, Relevant`}, so the full project would be a 4-class classification task. The typical motivating factor to structure the task in this manner is if the questions represent a rather different search intent than can be returned using the standard semantic search approach. The other primary reason is if you want to be able to assign a probability to the labels of the search task, which is not possible with the standard semantic search approach.

The core semantic search capabilities are already uniquely strong in Reexpress one without training for reranking, but reranking may be useful for the types of cases described above. As a general practice, we recommend first experimenting with the standard semantic search interface with your data, including in conjunction with constraining your data based on the additional criteria available in Explore->Select and Compare->Select, before collecting new data to train for reranking.

[3] The updatable `group` and `info` fields can be used as a memo to record notes as you explore your data.

# Discovery awaits.