tf.contrib.metrics.sparse_recall_at_top_k

```
sparse_recall_at_top_k(
    labels,
    top_k_predictions,
    class_id=None,
    weights=None,
    metrics_collections=None,
    updates_collections=None,
    name=None
)
```

Defined in `tensorflow/contrib/metrics/python/ops/metric_ops.py`.

Computes recall@k of top-k predictions with respect to sparse labels.

If `class_id` is specified, we calculate recall by considering only the entries in the batch for which `class_id` is in the label, and computing the fraction of them for which `class_id` is in the top-k `predictions`. If `class_id` is not specified, we'll calculate recall as how often on average a class among the labels of a batch entry is in the top-k `predictions`.

`sparse_recall_at_top_k` creates two local variables, `true_positive_at_<k>` and `false_negative_at_<k>`, that are used to compute the recall_at_k frequency. This frequency is ultimately returned as `recall_at_<k>`: an idempotent operation that simply divides `true_positive_at_<k>` by total (`true_positive_at_<k>` + `false_negative_at_<k>`).

For estimation of the metric over a stream of data, the function creates an `update_op` operation that updates these variables and returns the `recall_at_<k>`. Set operations applied to `top_k` and `labels` calculate the true positives and false negatives weighted by `weights`. Then `update_op` increments `true_positive_at_<k>` and `false_negative_at_<k>` using these values.

If `weights` is `None`, weights default to 1. Use weights of 0 to mask values.

Args:

- `labels`: `int64` `Tensor` or `SparseTensor` with shape [D1, ... DN, num_labels], where N >= 1 and num_labels is the number of target classes for the associated prediction. Commonly, N=1 and `labels` has shape [batch_size, num_labels]. [D1, ... DN] must match `top_k_predictions`. Values should be in range [0, num_classes), where num_classes is the last dimension of `predictions`. Values outside this range always count towards `false_negative_at_<k>`.
- `top_k_predictions`: Integer `Tensor` with shape [D1, ... DN, k] where N >= 1. Commonly, N=1 and top_k_predictions has shape [batch size, k]. The final dimension contains the indices of top-k labels. [D1, ... DN] must match `labels`.
- `class_id`: Integer class ID for which we want binary metrics. This should be in range [0, num_classes), where num_classes is the last dimension of `predictions`. If class_id is outside this range, the method returns NAN.
- `weights`: `Tensor` whose rank is either 0, or n-1, where n is the rank of `labels`. If the latter, it must be broadcastable to `labels` (i.e., all dimensions must be either `1`, or the same as the corresponding `labels` dimension).
- `metrics_collections`: An optional list of collections that values should be added to.
- `updates_collections`: An optional list of collections that updates should be added to.
- `name`: Name of new update operation, and namespace for other dependent ops.

Returns:

- `recall`: Scalar `float64` `Tensor` with the value of `true_positives` divided by the sum of `true_positives` and `false_negatives`.
- `update_op`: `Operation` that increments `true_positives` and `false_negatives` variables appropriately, and whose value matches `recall`.

Raises:

- `ValueError`: If `weights` is not `None` and its shape doesn't match `predictions`, or if either `metrics_collections` or `updates_collections` are not a list or tuple.

Last updated November 2, 2017.