Evaluation

Why?

Language technologies are designed to accomplish particular tasks, and as developers of these technologies, we need a way to measure their ability to satisfactorily perform the tasks for which they have been designed.

Of course, this is not the only factor which determines success. We may may also be interested in:

- cost, efficiency, scalability, compatibility,
- fault-tolerance,
- legal or ethical concerns,
- etc.

Outline for today

- The typology of evaluation (after Resnik & Lin 2010)
- Some common evaluation metrics
- Out-of-sample evaluation
- System comparison
- Performance bounds

The typology of evaluation

Local outline

- *automatic* vs. *manual* evaluation,
- *formative* vs. *summative* evaluation,
- *intrinsic* vs. *extrinsic* evaluation, and
- *component* vs. *end-to-end* evaluation.

Automatic vs. manual evaluation

Arguably, the most natural way to evaluate a system is to ask humans to assess its performance on some task. Such *manual evaluations* are considered the gold standard for certain tasks. Unfortunately, they may be often slow and/or expensive, and may pose substantial experimental design challenges.

Automatic evaluations depend on *evaluation metrics* which can be computed without further human intervention. Even when manual evaluations are the gold standard for a task, automatic "understudy" evaluation metrics enable rapid development.

Formative vs. summative evaluation

Formative evaluations compare a system's current performance to other (e.g., earlier) iterations of that system. They can be used to

- measure development progress,
- perform feature ablation, or
- optimize hyperparameters.

Summative evaluations compare competing systems.

"When the cook tastes the soup, that's formative; when the customer tastes the soup, that's summative."
(Resnik & Lin 2010: 274)

Intrinsic vs. extrinsic evaluation

Intrinsic evaluations measures a system's performance on the immediate task for which it is designed.

Extrinsic evaluations measure a system's contributions to some external (or *downstream*) task.

Component vs. end-to-end evaluation

Many language technologies consist of a "pipeline" of several interacting components. For example, a parser may depend on a part-of-speech tagger, which depends on a word tokenizer, which depends on a sentence boundary detector...

Component evaluations consider each component of the pipeline separately.

End-to-end evaluations focus on the final output of the pipeline.

Natural affinities

Resnik & Lin claim there are natural affinities between *automated*, *formative*, and *intrinsic* evaluations, and between *manual*, *summative*, and *extrinsic* evaluations.

For instance, an automatic "understudy" evaluation used to enable rapid system development is also a formative evaluation.

I would add that end-to-end evaluations and extrinsic evaluations are closely related.

Some common evaluation metrics

Local outline

For some types of tasks, like

- unstructured classification,
- structured classification, and
- sequence generation,

the choice of metric is clear. For others, it is governed by convention.

Unstructured classification metrics

accuracy: the percentage (or, the MLE probability) of correct classification.

For binary classification, there are many other possibilities.

The information retrieval contingency table

	True	**False**
True	True positive (TP)	False positive (FP)
False	False negative (FN)	True negative (TN)

Information retrieval metrics (1/)

The percentage of retrieved documents which are relevant is known as *precision* (or *positive predictive value*).

$$P = TP / (TP + FP)$$

The percentage of relevant documents which are successfully retrieved is known as *recall* (or *sensitivity*, or *true positive rate*):

$$R = TP / (TP + FN)$$

Precision fails to penalize false negatives; recall fails to penalize false positives.

Information retrieval metrics (2/)

To quantify the tradeoff between the two, it is common to use the harmonic mean of precision and recall, the *F-score* (or *F-measure*):

$$F = (2 \times P \times R) \,/\, (P + R)$$

NB: the harmonic mean of two positive numbers is always closer to smaller of the two, so if you want to maximize *F*-score, you should work harder on the lesser term.

Problem

Compute accuracy, precision, recall, and F-score for the following contingency table:

	True	**False**
True	10	2
False	5	20

$A = (TP + TN) / (TP + TN + FP + FN)$

$P = TP / (TP + FP)$

$R = TP / (TP + FN)$

$F = (2 \times P \times R) / (P + R)$

Solution

Compute accuracy, precision, recall, and F-score for the following contingency table:

	True	**False**
True	10	2
False	5	20

$A = .812$

$P = .666$

$R = .833$

$F = .741$

Precision and recall

Structured classification metrics

For tasks that extract "chunks" (like chunking, named entity recognition, etc.):

> Artists on its roster include [Hannah Diamond]$_{\text{per}}$, [GFOTY]$_{\text{per}}$, [Life Sim]$_{\text{org}}$, and [Danny L Harle]$_{\text{per}}$.

it is common to evaluate using the *F*-score over the retrieved chunks.

An implementation of this is included in the `nltk.chunk` API.

Sequence prediction metrics

For tasks that involve generating sequences of arbitrary length, it is common to use metrics based on *edit distance* (or *Levenshtein distance*), which counts the minimum number of "edits" (insertions, deletions and optionally, substitutions) needed to map the predicted sequence onto the ground truth sequence.

When edit distance is scaled by dividing edit distance by the length of the ground truth sequence, it is called *label error rate* (or *word error rate*, *phoneme error rate*, *character error rate* etc.).

Flexible Levenshtein distance in Python

Task-specific metrics

Certain tasks conventionally are evaluated using complex, task-specific metrics. E.g.:

- Labeled attachment accuracy (LAS), a metric for dependency parsing
- PARSEVEL, a metric for constituency parsing
- BLEU, an understudy metric for machine translation
- ROUGE, an understudy metric for automated summarization

Combined metrics

For some tasks, additional metrics can be created by propagating (or otherwise combining) low-level predictions to higher-level representations. E.g.:

- For POS tagging, we can derive *sentence accuracy* from token accuracy.
- For grapheme-to-phoneme conversion, we can derive *word error rate* from phoneme error rate.

Out-of-sample evaluation

Out-of-sample evaluation

Insofar as our goal is to measure generalization, it is *essential* that an evaluation be performed using disjoint training and test sets.

Evaluations performed on the training set—*resubstitution* evaluation—massively overestimates system performance.

"In its purest form…the test data should remain entirely untouched and unseen by the researcher or developer until system development is frozen just prior to evaluation."
(Resnik & Lin 2010: 278)

Notation

- G is the ground truth data
- S is a system with arbitrary parameters and hyperparameters
- $\mathcal{M}$ is an evaluation metric, a function with domain $G \times S$

WLOG, we assume that higher values of $\mathcal{M}$ indicate better performance.

Static partitioning

G is partitioned* into:

- Training set G_{train}
- Test (evaluation) set G_{test}

During training, we set the parameters of *S* so as to maximize $\mathcal{M}(G_{\text{train}}, S)$.

The figure of merit for *S* is then given by $\mathcal{M}(G_{\text{test}}, S)$.

*This may be a *random split* or conventionalized *standard split*.

Static partitioning with a development set

...

- Training set G_{train}
- Development (tuning) set G_{dev}
- Test (evaluation) set G_{test}

Set the parameters of S so as to maximize $\mathcal{M}(G_{\text{train}}, S)$ and the hyperparameters of S so as to maximize $\mathcal{M}(G_{\text{dev}}, S)$.

...

k-fold cross-validation

Instead of a single point estimate of $\mathcal{M}$, we draw k estimates $\mathcal{M}_1$ … $\mathcal{M}_k$.

G is partitioned into k "folds" such that G_i indicates the ith fold. Then for $i = \{1..k\}$:

- Let $G_{\text{test}} = G_i$ and $G_{\text{train}} = G - G_i$
- Set the parameters of S so as to maximize $\mathcal{M}(G_{\text{train}}, S)$
- Let $\mathcal{M}_i = \mathcal{M}(G_{\text{test}}, S)$

These estimates can then be aggregated statistically (e.g., with median or range).

When k is the number of samples, this is called *leave-one-out* (LOO) cross-validation.

k-fold cross-validation with a development set

...

Then for $i = \{1..k\}$:

- Let $G_{\text{test}} = G_i$, $G_{\text{dev}} = G_{i-1}$, and $G_{\text{train}} = G - \{G_{\text{test}}, G_{\text{dev}}\}$
- Set the parameters of S so as to maximize $\mathcal{M}(G_{\text{train}}, S)$ and the hyperparameters of S so as to maximize $\mathcal{M}(G_{\text{dev}}, S)$
- Let $\mathcal{M}_i = \mathcal{M}(G_{\text{test}}, S)$

...

Rules of thumb

- G_{train} is 70-80% of the labeled data
- G_{dev} and G_{test} are roughly the same size

The development-test set

In some formal settings (certain shared tasks, Kaggle competitions) a *development-test* (or *devtest*) set is held out for formative evaluations during system development before the test set has been made available.

System comparison

The standard system ranking procedure

Let S_1 and S_2 be systems trained as above, and let $\mathcal{M}_1$ and $\mathcal{M}_2$ be the associated figures of merit. Then we prefer S_1 to S_2 if and only if $\mathcal{M}_1 > \mathcal{M}_2$.

Problems with the standard ranking procedure

It treats $\mathcal{M}$ as an exact quantity instead of an estimate of a random variable. This may lead to *Type I error*, the error of falsely rejecting the null hypothesis that the systems being compared are equivalent.

In fact, many evaluation metrics have a known statistical distribution.

Review: the binomial distribution

$B(n, p)$ is a discrete probability distribution over the number of successes of n independent binary experiments which probability of success p.

A statistical model of relative accuracy (1/)

Let x be a Bernoulli random variable which takes on the value 1 when a test sample is correctly classified and 0 otherwise.

Then, accuracy is the maximum likelihood estimate of $p = P(x = 1)$ for a binomial random variable $B(N, p)$ where N is the number of test samples.

We can also compute variance, confidence intervals (Wilson 1927), etc.

A statistical model of relative accuracy (2/)

Now let $x_{1>2}$ be a Bernoulli random variable which has value 1 just when S_1 correctly classifies a test sample that S_2 misclassifies.

Then, there is a corresponding binomial random variable $B(n, p)$ where $p = \mathrm{P}(x_{1>2} = 1)$ and n is the number of test samples on which S_1 correctly classifies a sample that S_2 misclassifies or vice versa.

McNemar's test (Gillick & Cox 1989) adopts the null hypothesis that $p = .5$.*

*Normally, we use the two-sided, "mid-p" variant.

Other reference distributions

Many other common metrics and comparisons can be mapped onto known statistical distributions, and for those which cannot, a reference distribution can be estimated with *bootstrap resampling*.

Yet...

Dror et al. (2018) survey statistical practices in all long papers presented at the 2017 meeting of the ACL, and in that year's volume of *Transactions of the ACL*. They find:

- the majority of works do not use statistical tests for system comparison;
- many of those which do, do not use appropriate statistical tests; and
- many of those which do, do not report the test(s) used.

This is a chance to distinguish yourself.

Performance bounds

Inter-annotator agreement handout

Oracle upper bound

Another type of upper bound is provided by constructing an *oracle* system which has access to ground truth labels during inference.

> Imagine we are doing a system comparison of several part-of-speech taggers. There are various ways we might combine these taggers to create an *ensemble* tagger. There is no foolproof way to select from amongst the predictions of the systems that make up the ensemble…
>
> ...except by using the ground truth labels. The *oracle ensemble* predicts the correct tag so long as at least one of the taggers in the ensemble does.

Baseline lower bounds

It is often desirable to compute a *baseline* lower bound.

For formative evaluations of unstructured classification tasks, one common absolute baseline is a model which predicts the most common label for every example.

For summative evaluations, a tighter baseline is obtained from prior models, or prior literature that reports results on the same data set.

2x2 contingency table tutorial

Final thoughts

- An ideal metric has lots of headroom; it becomes less informative near ceiling
- From an machine learning perspective, it is good for there to be a close connection between the training objective and the metric
- From an evaluation perspective, this is much less clear; it tends to expose the limitations and corner cases of the metric.