Evaluation measures in NLP

Zdeněk Žabokrtský

October 26, 2022

Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

EUROPEAN UNION
European Structural and Investment Fund
Operational Programme Research, Development and Education

unless otherwise stated

Outline

Evaluation goals and basic principles

Selected good practices in experiment evaluation

Selected task-specific measures

Big picture

Final remarks

Evaluation goals and basic principles

Basic goals of evaluation

- The goal of NLP evaluation is to measure one or more qualities of an algorithm or a system.
- A side effect: Definition of proper evaluation criteria is one way to specify precisely an NLP problem (don't underestimate this!).
- Need for evaluation metrics and evaluation data (actually evaluation is one of the main reasons why we need language data resources).

Automatic vs. manual evaluation

- automatic
  - comparing the system's output with the gold standard output and evaluate e.g. the percentage of correctly predicted answers
  - the cost of producing the gold standard data...
  - ...but then easily repeatable without additional cost
- manual
  - manual evaluation is performed by human judges, which are instructed to estimate the quality of a system, based on a number of criteria
  - for many NLP problems, the definition of a gold standard (the "ground truth") can prove impossible (e.g., when inter-annotator agreement is insufficient)

Intrinsic vs. extrinsic evaluation

- Intrinsic evaluation
  - considers an isolated NLP system and characterizes its performance mainly
- Extrinsic evaluation
  - considers the NLP system as a component in a more complex setting

The simplest case: accuracy

- accuracy – just a percentage of correctly predicted instances

$$accuracy = \frac{correctly\ predicted\ instances}{all\ instances} \times 100\%$$

- correctly predicted = identical with human decisions as stored in manually annotated data
- an example – a part-of-speech tagger
  - an expert (trained annotator) chooses the correct POS value for each word in a corpus,
  - the annotated data is split into two parts
  - the first part is used for training a POS tagger
  - the trained tagger is applied on the second part
  - POS accuracy = the ratio of correctly tokens with correctly predicted POS values

Selected good practices in experiment evaluation

Experiment design

- the simplest loop:
  1. train a model using the training data
  2. evaluate the model using the evaluation data
  3. improve the model
  4. goto 1
- What's wrong with it?

A better data division

- the more iterations of the loop, the more fuzzy distinction between training and evaluation phases
- in other words, evaluation data effectively becomes training data after repeated evaluations (the more evaluations, the worse)
- but if you "spoil" all the data by using it for training, then there's nothing left for an ultimate evaluation
- this problem should be foreseen: divide the data into three portions, not two:
  - training data
  - development data (devset, devtest, tuning set)
  - evaluation data (etest) – to be used only once (in a very long time)
- sometimes even more complicated schemes are needed (e.g. for choosing hyperparameter values)

K-fold cross-validation

- used especially in the case of very small data, when an actual train/test division could have huge impact on the measured quantities
- averaging more training/test divisions, usually K=10
  1. partition the data into K roughly equally-sized subsamples
  2. perform cyclically K iterations:
     - use $K-1$ subsamples for training
     - use 1 sample for testing
  3. compute arithmetic average value of the K results
  4. more reliable results, especially if you face very small or in some sense highly diverse data

Interpreting the results of evaluation

- You run an experiment, you evaluate it, you get some evaluation result, i.e. some number ...
- ...is the number good or bad?
- No easy answer
- However, you can interpret the results with respect to expected upper and lower bounds.

Lower and upper bounds

- the fact that the domain of accuracy is 0–100% does not imply that any number within this interval is a reasonable result
- the performance of a system under study is always expected to be inside the interval given by
  - lower bound – result of a baseline solution (less complex or even trivial system the performance of which is supposed to be easily surpassed)
  - upper bound – result of an oracle experiment, or level of agreement between human annotators

Baseline

- usually there's a sequence of gradually more complex (and thus more successful) methods
- in the case of classification-like tasks:
  - random-choice baseline
  - most-frequent-value baseline
  - ...
  - some intermediate solution, e.g. a "unigram model" in POS tagging $argmax(P(tag|word))$
  - ...
  - the most ambitious baseline: the previous state-of-the-art result
- in structured tasks:
  - again, start with predictions based on simple rules
  - examples for dependency parsing
    - attach each node below its left neighbor
    - attach each node below the nearest verb
    - ...

Oracle experiment

- the purpose of oracle experiments – to find a performance upper bound (from the viewpoint of our evaluated component)
- oracle
  - an imaginary entity that makes the best possible decisions
  - however, respecting other limitations of the experiment setup
- naturally, oracle experiments make sense only in cases in which the "other limitations" imply that even oracle's performance is less than 100 %
- example – oracle experiment in POS tagging
  - for each word, collect all its possible POS tags from a hand-annotated corpus
  - on testing data, choose the correct POS tag whenever it is available in the set of possible tags for the given word

Noise in annotations

- recall that "ground truth" (=correct answers for given task instances) is usually delivered by human experts (annotators)
- So let's divide data among annotators, collect them back with their annotations, and we are done ...No, this is too naive!
- Annotation is "spoiled" by various sources of inconsistencies:
  - It might take a while for an annotator to grasp all annotation instructions and become stable in decision making.
  - Gradually gathered annotation experience typically leads to improvements of (esp. more detailed specification of) annotation instructions.
  - But still, different annotators might make different decisions, even under the same instructions.
  - ...and of course many other errors due to speed, insufficient concentrations, working ethics issues etc., like with any other work.
- It is absolutely essential to quantify inter-annotator agreements.

Inter-annotator agreement (IAA) measure

- IAA is supposed to tell us how good human experts perform when given a specific task (to measure the reliability of manual annotations)
- In the simplest case: accuracy measured on data data from one annotator, with the other annotator being treated as a virtual gold standard (symmetric)
- Slightly more complex: using F-measure instead of accuracy (still symmetric if F1).
- But is e.g. IAA=0.8 good enough, or not?
- We should consider a baseline for IAA (which is the agreement level gained without any real decision making).

Inter annotator agreement – agreement by chance

Example:

- two annotators making classifications into two classes, A and B
- 1st annotator: 80% A, 20% B
- 2nd annotator 85% A, 15% B
- probability of agreement by chance: 0.8*0.85 + 0.2*0.15 = 71%

desired measure: 1 if they agree in all decisions, 0 if their agreement is equal to agreement by chance

Cohen's Kappa

- takes into account the agreement by chance

$$\kappa = \frac{P_a - P_e}{1 - P_e}$$

- P_a = relative observed agreement between two annotators (i.e., probability of agreement)
- P_e = probability of agreement by chance
- scale from -1 to $+1$ (negative kappa unexpected but possible)
- interpretation still unclear, but at least we abstracted from the by-chance agreement baseline
- conventional interpretation: ...0.40-0.59 weak agreement, 0.60-79 moderate agreement, 0.80-0.90 strong agreements ...

Rounding

- In most cases, we present results in the decimal system.
- Inevitably, we express two quantities when presenting any single number:
  - the value itself
  - and our certainty about the value, rendered by using a specific number of significant digits.
- **Writing more digits** than justified by the experiment setup **is a bad habit!**
- You are misleading the reader if you say that the error rate of your system is 42.8571% if it has made 3 errors in 7 task instances (you indicate more more certainty about the result than justified).
- In short, the number of significant digits is linked to experiment setting and reflects its result uncertainty.

Basic rules for rounding

Actually very simple:

- Multiplication/division - the number of significant digits in an answer should equal the least number of significant digits in any one of the numbers being multiplied/divided.
- Addition/subtraction - the number of decimal places (not significant digits) in the answer should be the same as the least number of decimal places in any of the numbers being added or subtracted.

Selected task-specific measures

Accuracy revisited

$$accuracy = \frac{correctly\ predicted\ instances}{all\ instances} \times 100\%$$

Easy if

- the number of task instances is known
- there is exactly one correct answer for each instance
- our system gives exactly one answer for each instance
- all errors are equally wrong

But what if not?

Precision and recall

- precision – if our systems gives a prediction, what's its avarage quality

$$precision = \frac{correct\ answers\ given}{all\ answers\ given} \times 100\%$$

- recall – what avergage proportion of all possible correct answers is given by our system

$$recall = \frac{correct\ answers\ given}{all\ possible\ correct\ answers} \times 100\%$$

Precision and recall – new issues

- computing precision and recall $\rightarrow$ we are in 2D now
- but in most cases we want to be able order all systems along a single scale
- in simple words: in 2D its sometimes hard to say what is better and what is worse
- example: is it better to have a system with P=0.8 and R=0.2, or P=0.9 and R=0.1 ?
- thus we want to get back to 1D again
- a possible solution: F-measure

Get back to 1D: F-measure

- F-measure = weighted harmonic mean of P and R
- (note: if two quantities are to be weighted, we need just 1 weighting parameter X, and the other one can be computed 1-X)
- $$F_\beta = \frac{(\beta^2 + 1) \cdot precision \cdot recall}{\beta^2 \cdot precision + recall}$$
- exercise: show how did this formula came from the formula for the harmonic mean?
- usually weighted evenly (=no β needed):

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Precision at K

- in some situations, recall is close-to-impossible to estimate
- example: how many relevant web pages for a given query are there on the Internet?
  - impossible to hand-check
  - useless anyway, because no users are interested in reading them all
- precision at 10 (P10 for short) in Information retrieval
  - proportion of top 10 documents that are relevant
  - disadvantage: disregards ordering of hits in those top 10 documents

Word error rate (WER)

- common metric for speech recognitions
- typically, the number of recognized words can be quite different from the number of true words

$$WER = \frac{S + D + I}{S + D + C} \tag{1}$$

- S – substituted words, D – deleted words, I – inserted words, C – correct words
- exercise: one substitution is equivalent to one deletion plus one insertion, what should we do with that?

BLEU (bilingual evaluation understudy)

$$BLEU = BP \cdot \exp\left(\frac{1}{4}\sum_{n=1}^{4} \log p_n\right)$$

$$BP = \min\left(1, \exp\left(1 - \frac{r}{c}\right)\right)$$

BP = brevity penalty (multiplicative!)
p_n = n-gram precision (ref-clipped counts)
r = total length (#words) of the reference
c = total length of the candidate translation

- a common measure in machine translation
- measures similarity between a machine's output and a human translation
- criticism
  - todo

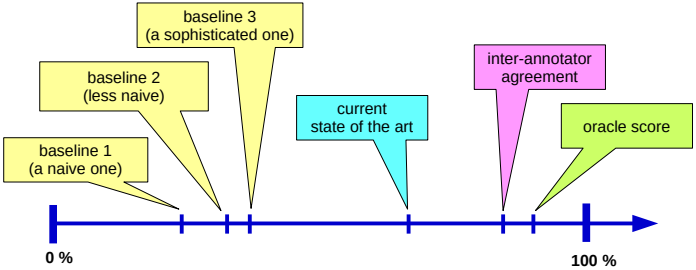
Global view on NLP evaluation: a compromise is often needed

- more complex evaluation measures are often designed as a compromise (a trade-off) between two or more criteria pushing in different directions
  - precision against recall in F-measure
  - n-gram precision against brevity penalty in BLEU
  - in manual evaluation of machine translation: fluency against adequacy

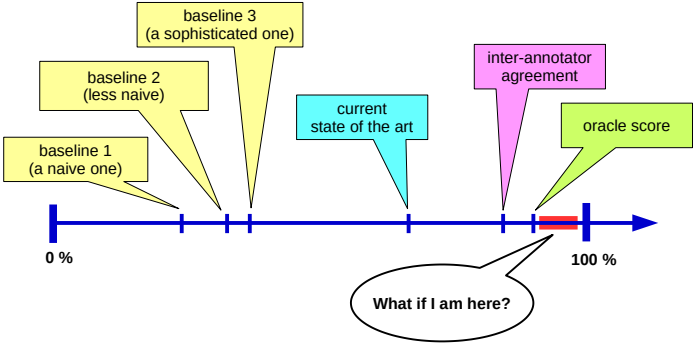
Big picture

Evaluation needs a larger context

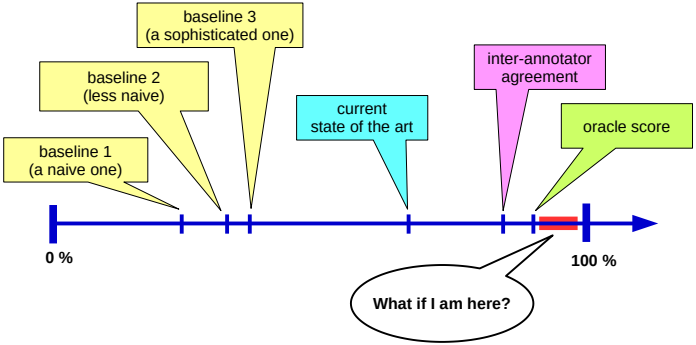
- An example evaluation setup before you start measuring the performance of your new and shiny method developed for a given NLP task, using a given dataset and a given evaluation measure:



Evaluation needs a larger context, cont.

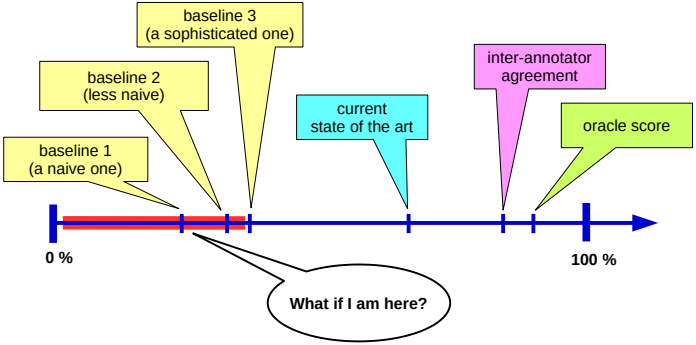


Evaluation needs a larger context, cont.

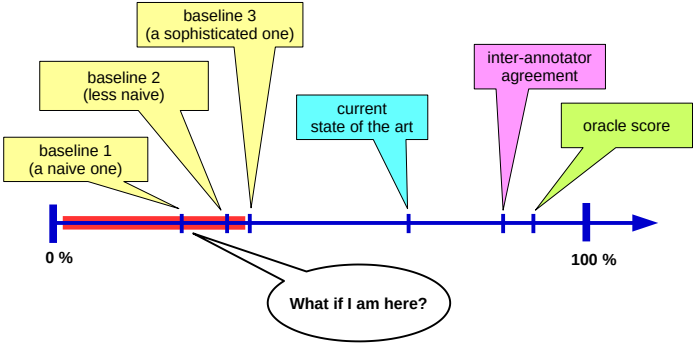


- Comments: outperforming an oracle is impossible by definition. Search for a bug in your evaluation script.

Evaluation needs a larger context, cont.

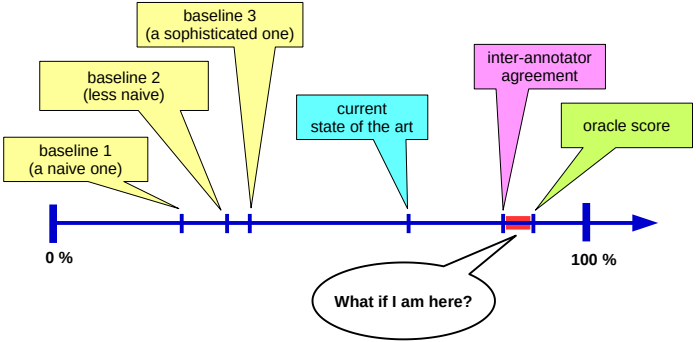


Evaluation needs a larger context, cont.

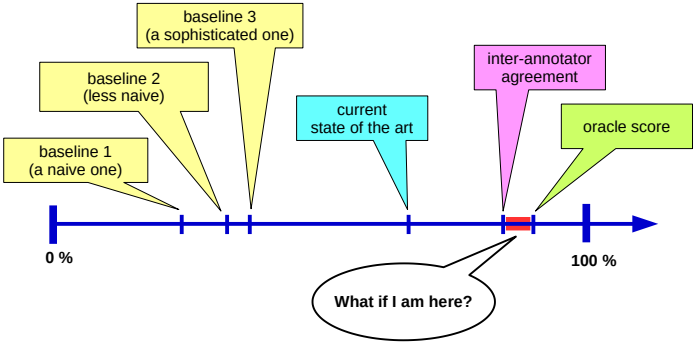


- Comments: Bad luck. Maybe the structure of your ML model does not fit the task well, and/or its hyper-parameters are not set properly, and/or the performance is spoiled by underfitting or overfitting. Revise the code to avoid bugs and apply standard ML diagnostics (e.g. examine learning curves). Before beating the baselines, it's unfortunately not worth publishing.

Evaluation needs a larger context, cont.

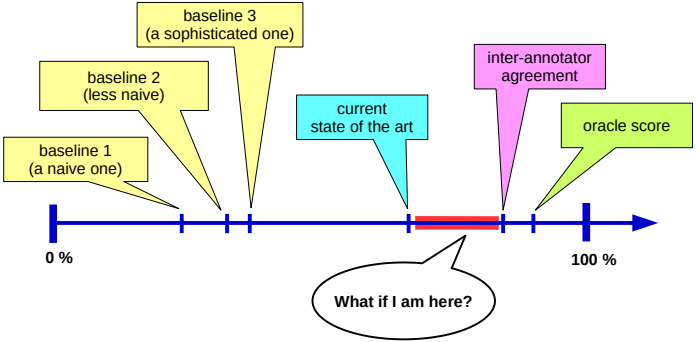


Evaluation needs a larger context, cont.

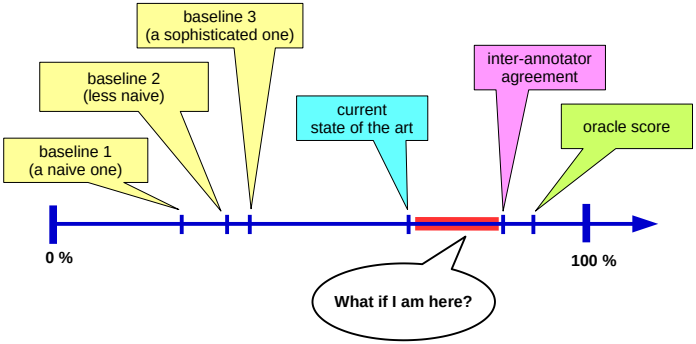


- Comments: This is possible but suspicious, as humans still outperform machines in quality measures in most NLP tasks nowadays. Didn't you e.g. mix training and evaluation data by mistake?

Evaluation needs a larger context, cont.

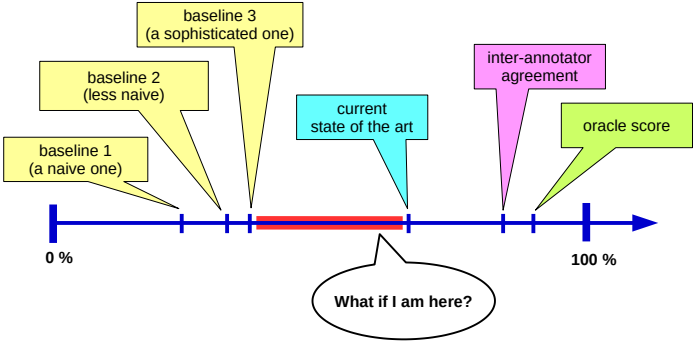


Evaluation needs a larger context, cont.

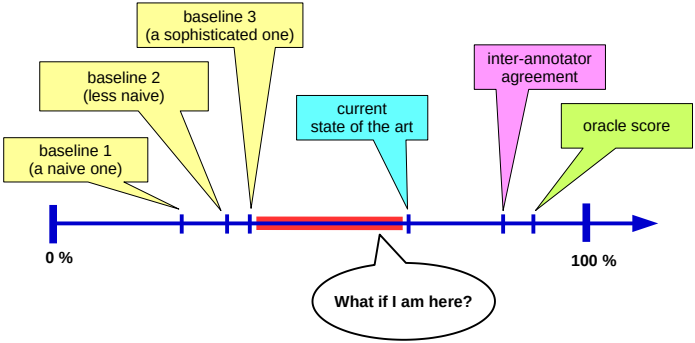


- Comments: congrats, it seems you've just established a new state of the art! Hurry up for publishing it! But first double-check that the whole experimental setup is correct and fair (again, didn't some pieces of information from the evaluation portion penetrate the training data? Didn't you use the evaluation data too many times?)

Evaluation needs a larger context, cont.



Evaluation needs a larger context, cont.



- Comments: this is by far the most usual experimental outcome. You outperformed all the baselines, which is nice, but you did not beat the current champion, which is ...usual. Depending on details, your approach might still be worth using by others, e.g. because of its speed, its robustness in under-resourced scenarios, or its freer license policy.

Final remarks

Frequent problems with golden-data-based evaluation in NLP

- Unclear "ground truth" – what if even human annotators disagree?
- Even worse, sometimes the search space is so huge that creating reasonably represenative hand-annotated data is virtually impossible (e.g. a path through a man-machine dialogue, if more dialogue turns are considered)
- Unclear whether all errors should be trated equally (e.g. attaching a verb's argument in dependency parsing seems more important than attaching a punctuation mark); if weighting is needed, then we always risk arbitrariness.
- A rather recent problem: for some tasks, the quality of state-of-the-art solutions is already above that of average annotators (i.e., even hand-annotated gold data might not be gold enough)

Take-home messages

- using data for evaluation – perhaps the most important role of data in NLP
- the existence of data allows exact measurement, like e.g. in physics,
- however, unlike in physics, evaluation measures in NLP are almost always based on various assumptions (even using plain percentage in the simplest cases implies that we assume that all errors are born equally severe, which is a unrealistic)
- unlike in physics, evaluation results achieved by the same methods might differ considerably across datasets/genres/languages...
- because of this limited interpretability, in NLP evaluation it's not enough just to present a final value of the measured quantity – you should always make it anchored in a bigger picture!