Natural Language Processing Introduction

Unit 2: classifcation evaluation

February 2020

Classification evaluation

Concept

Informally speaking classification evaluation is an important step in every classifcation system, which tell us how well our systems classify new data.

Why is important?

- Give a meassure of the reliability of the system
- Important dimension at system design phase
- Metric to messure the improvement of the system
- Give valuable insights about tha data and training quality

Evaluating classification

To introduce the methods for evaluating text classification, let's first consider some simple binary detection tasks. For example, in spam detection, our goal is to label every text as being in the spam category ("positive") or not in the spam category ("negative").

Evaluating classification: gold labels

We also need to know whether the email is actually spam or not, i.e. the human-defined labels for each document that we are trying to match. We will refer to these human labels as the gold labels.

Precission

When we are having biased classes, say 99 % of the examples are not spam and 1 % is spam, we can build a dummy classificator as follows:

Input: x example to be classified as positive or negative
Output: y boolean *True* if the class is positive *False* otherwise

---

1: **return** *False*

Such classifier will have a 99 % of accuracy!.

Precission

That's why instead of accuracy we generally turn to two other metrics: precision and recall. Precision measures the percentage of the items that the system detected that are in fact positive (i.e., are positive according to the human gold labels). Precision is defined as:

$$P = \frac{true\ \ positives}{true\ \ positives + false\ \ positives}$$

Recall

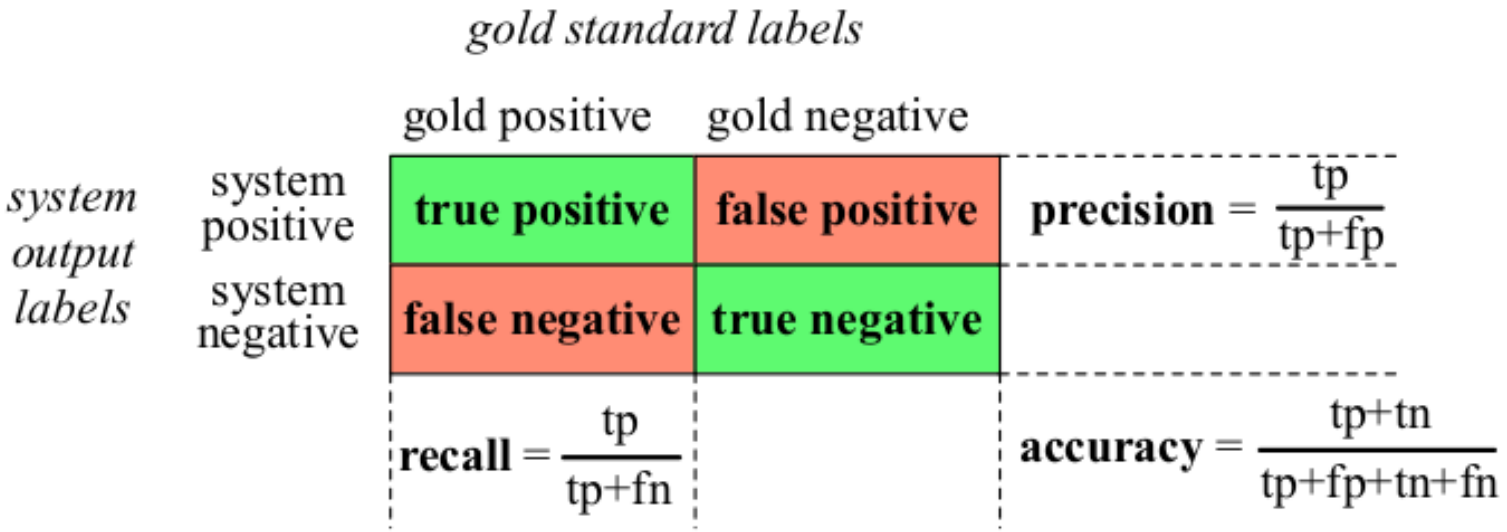
Recall measures the percentage of items actually present in the input that were correctly identified by the system. Recall is defined as:

$$R = \frac{true\ \ positives}{true\ \ positives + false\ \ negatives}$$

Evaluating classification: contingency table

To evaluate any system for detecting things, we start by building a contingency table. Each cell labels a set of possible outcomes. In the spam detection case, for example, true positives are documents that are indeed spam (indicated by human-created gold labels) and our system said they were spam. False negatives are documents that are indeed spam but our system labeled as non-spam.

Contingency table example



F score

Precision and recall will help solve the problem with the useless "nothing is pie" classifier. This classifier, despite having a fabulous accuracy of 99 %, has a terrible recall of 0, for instance if we have 1000 examples (since there are no true positives, and 10 false negatives, the recall is 0/10).

F score

There are many ways to define a single metric that incorporates aspects of both precision and recall. The simplest of these combinations is the F measure, defined as:

$$F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P+R}$$

The β parameter differentially weights the importance of recall and precision, based perhaps on the needs of an application. Values of $\beta > 1$ favor recall, while values of $\beta < 1$ favor precision.

F_1 score

F measure comes from a weighted harmonic mean of precision and recall. The harmonic mean of a set of numbers is the reciprocal of the arithmetic mean of reciprocals, where $\beta = 1$ we obtain the F_1 score defined as:

$$F_1 = \frac{2PR}{P+R}$$

Multiple classes: any of

There are two kinds of multiclass classification tasks. In any of or multi label classification, each document or item can be assigned more than one label. We can solve any of classification by building separate binary classifiers for each class c, trained on positive examples labeled c and negative examples not labeled c.

Multiple classes: multinomial classification

More common in language processing is one-of or multinomial classification, in which the classes are mutually exclusive and each document or item appears in exactly one class. Here we again build a separate binary classifier trained on positive examples from c and negative examples from all other classes.

Microaveraging

In order to derive a single metric that tells us how well the system is doing, we can combine these values in two ways. In macroaveraging, we compute the performance for each class, and then average over classes. In microaveraging, we collect the decisions for all classes into a single contingency table, and then compute precision and recall from that table.

Confussion matrix

Confusion matrix for a three class categorization task, showing for each pair of classes (c_1 , c_2), how many documents from c_1 were incorrectly assigned to c_2:

		gold labels urgent	normal	spam	
system output	urgent	8	10	1	**precision**$_u$= $\frac{8}{8+10+1}$
	normal	5	60	50	**precision**$_n$= $\frac{60}{5+60+50}$
	spam	3	30	200	**precision**$_s$= $\frac{200}{3+30+200}$
		recall$_u$ = $\frac{8}{8+5+3}$	**recall**$_n$ = $\frac{60}{10+60+30}$	**recall**$_s$ = $\frac{200}{1+50+200}$	

Contingency table example

Separate contingency tables for the 3 classes:

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

$$\text{microaverage precision} = \frac{268}{268+99} = \mathbf{.73}$$

$$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = \mathbf{.60}$$

Train, validation and test sets

The training and testing procedure for text classification splits the data in 3 separate sets:

- Train set
- Validation (also called development test set or devset)
- Test set

We use the training set to train the model, then use the validation set to perhaps tune some parameters, and in general decide what the best model is. Once we come up with what we think is the best model, we run it on the (unseen) test set to report its performance.

Cross validation I

While the use of a devset avoids overfitting the test set, having a fixed training set, devset, and test set creates another problem: in order to save lots of data for training, the test set (or devset) might not be large enough to be representative.

Cross validation II

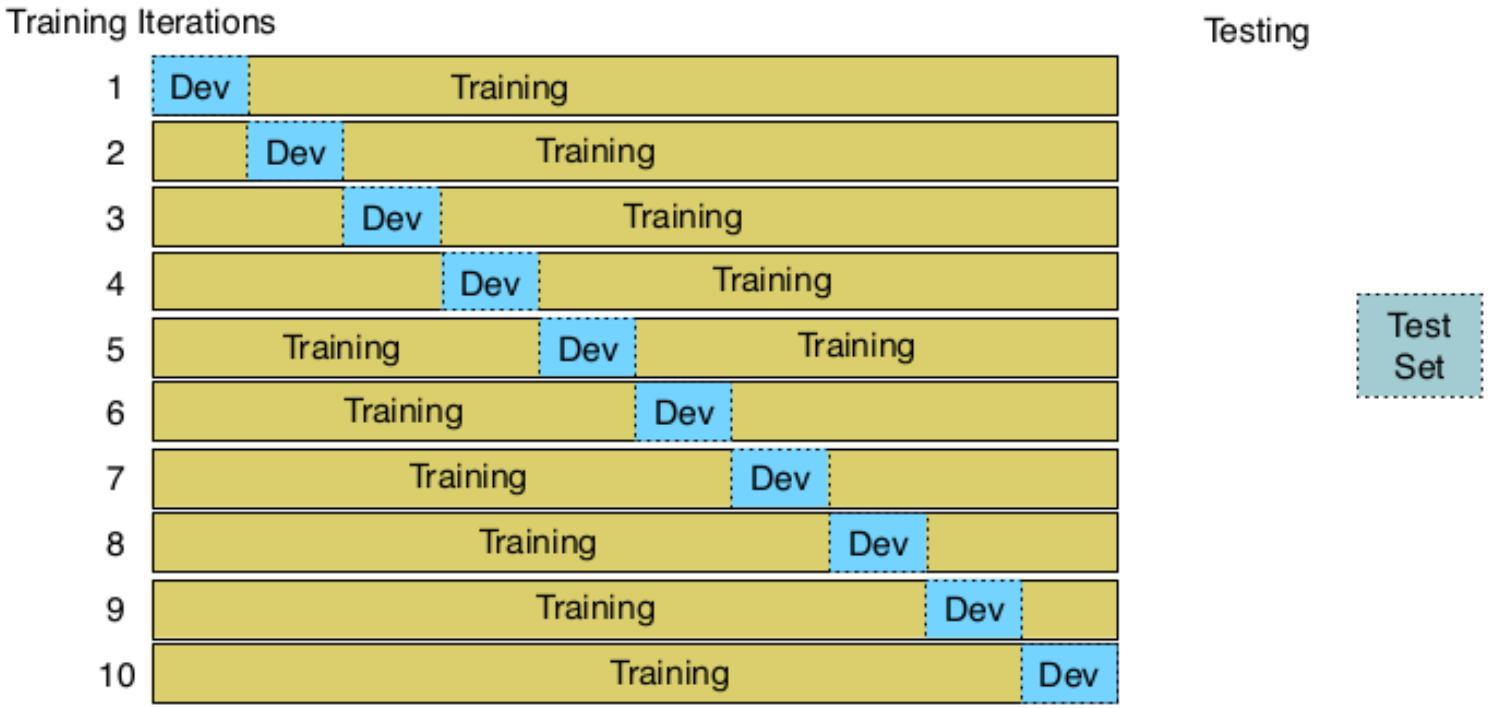
It would be better if we could somehow use all our data both for training and test. We do this by cross-validation: we randomly choose a training and test set division of our data, train our classifier, and then compute the error rate on the test set. Then we repeat with a different randomly selected training set and test set. We do this sampling process k times and average these k runs to get an average error rate.
This is called k fold cross validation.

Cross validation III

The only problem with cross-validation is that because all the data is used for testing, we need the whole corpus to be blind; we can't examine any of the data to suggest possible features and in general see what's going on. But looking at the corpus is often important for designing the system. For this reason, it is common to create a fixed training set and test set, then do k fold cross-validation inside the training set, but compute error rate the normal way in the test set

Cross validation IV

10 folds cross validation:



Assignments

Assignment 7: Build a sentiment analysis system using the twitter sentiemnt analysis database and evaluate your classffication:

- Implement a simple classifer using logistic regression (you can use sklearn class)
- Perform the trainin using a 4-folds cross validation strategy
- Report your Precission, Recall and F1 scores
- Write a Jupyter Notebook explaining your code

Let's code

References

[1] Jurafsky D., Martin J.: Speech and Language Processing 2nd. ed. (2009).

[2] Bird S., Klein E., Loper, E.: Natural Language Processing with Python, (2009). ISBN: 978-0-596-51649-9

[3] Indurkhya N., Damerau F. Handbook of Natural Language Processing, Second Edition (2010). ISBN: 978-1-4200-8593-8

[4] Kao, A., Poteet S. Natural Language Processing and Text Mining (2007). ISBN: 78-1-84628-175-4

[5] Sipser, M. Introduction to the theory of computation (2013). ISBN: 978-1-133-18779-0

[6] https://www.sketchengine.eu/corpora-and-languages/corpus-types/