Macro- and micro-averaged evaluation measures [[BASIC DRAFT]]

Vincent Van Asch

September 9, 2013

Abstract

This is a short paper introducing pitfalls when implementing averaged scores. Although, it is common to compute averaged scores, it is good to specify in detail how the scores are computed.

Contents

1 Introduction

Different evaluation measures are available when computing evaluation scores for classification. van Rijsbergen (1975) specifies recall (R) and precision (P) for information retrieval purposes. van Rijsbergen (1975) also introduces *effectiveness*, which can be rewritten in the more commonly known F-score or F-measure.

2 Basic measures

	True label A	True not A
Predicted label A	true positive (tp)	false positive (fp)
Predicted not A	false negative (fn)	true negative (tn)

Table 1: Performance table for instances labeled with a class label A .

Given a Table 1, one can compute:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{1}$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{2}$$

$$\text{F-score}(\beta) = \frac{(1+\beta^2)PR}{\beta^2 P + R} \tag{3}$$

Most commonly, β is taken to be 1.

You can also express the F-score in terms of true positives, false positives and false negatives:

$$\text{F-score}(\beta) = \frac{(1+\beta^2)\ tp}{(1+\beta^2)\ tp + \beta^2\ fp + fn} \tag{4}$$

You can compute the accuracy for class A:

$$\begin{aligned} \text{accuracy} &= \frac{\text{tp}}{N_A} \\ &\text{with} \\ \text{tp} &= \text{true positive for class A} \\ N_A &= \text{the total number of instances of class A in test} \end{aligned} \tag{5}$$

This definition equals recall for class A. Indeed, per class c, N_c equals $tp_c + fn_c$. When accuracy is reported in the literature, it is often computed as follows:

$$\begin{aligned} \text{accuracy} &= \frac{\sum_c^C \text{tp}_c}{N} \\ &\text{with} \\ \text{tp}_c &= \text{true positives for class c} \\ C &= \text{the number different classes in test} \\ N &= \text{the total number of instances in test} \end{aligned} \tag{6}$$

Which is equal to micro-averaged recall and for this reason accuracy is not mentioned further in this paper. In a binary situation, accuracy also boils down to:

$$\begin{aligned} \text{accuracy} &= \frac{\text{tp} + \text{tn}}{N} \\ &\text{with} \\ &\text{tp} = \text{true positives for class A} \\ &\text{tp} = \text{true negatives for class A} \\ &N = \text{the total number of instances in test} \end{aligned} \tag{7}$$

When classification uses multiple class labels, one can produce a table like Table 1 for each class label. Precision, recall, and F-score can be computed for each class label.

False positive rate (FPR), area-under-the-curve (AUC) and many other measures are not discussed in this paper.

3 Averaging single-label scores

When multiple class labels are to be retrieved, averaging the evaluation measures can give a view on the general results. There are two names to refer to averaged results: micro-averaged and macro-averaged results.

3.1 Macro-averaged measure

The macro-averaged results can be computed as indicated by:

> $L = \{\lambda_j : j = 1...q\}$ is the set of all labels. [...] Consider a binary evaluation measure $B(tp, tn, fp, fn)$ that is calculated based on the number of true positives (tp), true negatives (tn), false positives (fp) and false negatives (fn). Let tp_λ, fp_λ, tn_λ and fn_λ be the number of true positives, false positives, true negatives and false negatives after binary evaluation for a label λ. [...]

$$B_{macro} = \frac{1}{q} \sum_{\lambda=1}^{q} B(tp_\lambda, fp_\lambda, tp_\lambda, fn_\lambda) \tag{8}$$

(Tsoumakas et al., 2010)

The only possible ambiguity coming from the set of labels. Most commonly, the set of labels in the training data is used. This set may differ from the set of labels in the test data if there is no instance for a given class label in the test set.

For Table 2, the macro-averaged precision is $\frac{1}{2}\left(\frac{10}{10+10} + \frac{90}{90+10}\right) = 0.7$.

label	tp	fp	fn	precision	recall
c_1	10	10	10	0.5	0.5
c_2	90	10	10	0.9	0.9
total	100	20	20		
macro-averaged				0.7	0.7
micro-averaged				0.83	0.83

Table 2: Positive/negative counts for 2 class labels. All instances have only one class label.

On the difference between macro and micro-averaging:

> Macroaveraging gives equal weight to each class, whereas microaveraging gives equal weight to each per-document classication decision. Because the F1 measure ignores true negatives and its magnitude is mostly determined by the number of true positives, large classes dominate small classes in microaveraging. In the example [Table 2], microaveraged precision (0.83) is much closer to the precision of c_2 (0.9) than to the precision of c_1 (0.5) because c_2 is five times larger than c_1. Microaveraged results are therefore really a measure of effectiveness on the large classes in a test collection. To get a sense of effectiveness on small classes, you should compute macroaveraged results (Manning et al., 2008).

3.2 Micro-averaged measure

A micro-averaged can be computed as follows (Sebastiani, 2002; Manning et al., 2008; Tsoumakas et al., 2010):

$$B_{micro} = B\left(\sum_{\lambda=1}^{q} tp_\lambda, \sum_{\lambda=1}^{q} fp_\lambda, \sum_{\lambda=1}^{q} fn_\lambda, \sum_{\lambda=1}^{q} tn_\lambda\right) \tag{9}$$

For precision and recall in Table 2, this is $\frac{100}{100+20} = 0.83$. Again, ambiguity comes from which set of labels to take, commonly the class labels from the training data are used.

3.3 Label-frequency-based micro-averaged

Daelemans et al. (2010) introduce a second manner to compute a micro-averaged score:

$$B_{lfb} = \sum_{\lambda=1}^{q} f_\lambda B(tp_\lambda, fp_\lambda, tp_\lambda, fn_\lambda) \tag{10}$$

with f_λ the relative frequency of the class labels in the training set:

$$\begin{aligned} c_\lambda &= \text{number of instances in training with class label } \lambda \\ n &= \text{number of instances in training} \\ f_\lambda &= \frac{c_\lambda}{n} \end{aligned}$$

We call this manner of computation, the label-frequency-based micro-averaged scores of lfb-micro-averaged scores. More information when the frequencies f_λ are token from the test data rather than the training data can be found in Section 3.3.2.

3.3.1 Equivalent *micro* and *lfb*

$recall_{micro} = recall_{lfb}$ when relative frequency of the class labels for each label is the same in test and training set.[1]

$precision_{micro} = precision_{lfb}$ is only true when an additional condition holds: it should be the case that for each label $fp_\lambda = fn_\lambda$.

The validity of this assertion can be proven starting from the definition of $precision_{lfb}$.

In the situation of one-class per instance, the following equations hold:

[1]Other solutions to make $recall_{micro} = recall_{lfb}$ can emerge from the actual data. We only discuss a more general case.

$$\sum_{\lambda=1}^{q} tp_\lambda + \sum_{\lambda=1}^{q} fp_\lambda = \sum_{\lambda=1}^{q} tp_\lambda + \sum_{\lambda=1}^{q} fn_\lambda = N = \frac{1}{a}n \tag{11}$$

with $a > 0$, a factor linking the number of instances in training (n) to the number of instances in test (N). If $f_\lambda^{test} = f_\lambda^{train}$, then it also holds for the number of instances in training with label λ, i.e. c_λ, that

$$c_\lambda = a(tp_\lambda + fn_\lambda) \tag{12}$$

Using the definition of $precision_{lfb}$:

$$\begin{aligned} precision_{lfb} &= \sum_{\lambda=1}^{q} \frac{c_\lambda}{n} \cdot \frac{tp_\lambda}{tp_\lambda + fp_\lambda} \\ &= \sum_{\lambda=1}^{q} \frac{a(tp_\lambda + fn_\lambda)}{a\big(\sum_{\lambda=1}^{q} tp_\lambda + \sum_{\lambda=1}^{q} fp_\lambda\big)} \cdot \frac{tp_\lambda}{tp_\lambda + fp_\lambda} \\ &= \frac{1}{\sum_{\lambda=1}^{q} tp_\lambda + \sum_{\lambda=1}^{q} fp_\lambda} \sum_{\lambda=1}^{q} \frac{tp_\lambda + fn_\lambda}{tp_\lambda + fp_\lambda} \cdot tp_\lambda \end{aligned} \tag{13}$$

It can be seen that this reduces to the definition of $precision_{micro}$ if the extra condition holds $fp_\lambda = fn_\lambda$:[2]

$$precision_{lfb} = \frac{\sum_{\lambda=1}^{q} tp_\lambda}{\sum_{\lambda=1}^{q} tp_\lambda + \sum_{\lambda=1}^{q} fp_\lambda} \tag{14}$$

Similarly, it can be proven that the extra condition is not needed for recall. F-score is the harmonic mean of precision and recall and the value difference for *micro* and *lfb* thus depends on the the value difference of the precision.

The fundamental difference between *micro* and *lfb* being that the former does not use any information from the training data (or assumes that the relative label frequencies in test and training are the same).

A disadvantage of *micro* is that for the frequent single-label per instance problems: $precision_{micro} = recall_{micro}$. So, it is not possible to tune a system such that it has more averaged recall at the cost of averaged precision and vice versa using this evaluation score.

[2]The case $fp_\lambda = fn_\lambda$ is the most interesting solution of $\sum_{\lambda=1}^{q} \frac{fn_\lambda - fp_\lambda}{tp_\lambda + fp_\lambda} tp_\lambda = 0$. For all other solutions of this equation, $precision_{micro}$ also equals $precision_{lfb}$.

3.3.2 Label-frequency-based using test data frequencies

When the frequencies f_λ in Formula (10) are taken from the test data instead of the training data. The following scores are obtained:

- $recall_{lfb}$ becomes equal to $recall_{micro}$ (as seen in the previous section),
- $precision_{lfb}$ remains different from $precision_{micro}$ (except for some specific cases).

Because $recall_{lfb}$ and $precision_{lfb}$ are different, using the test data frequencies still offers the possibility to tune a system using lfb-micro-averaged precision and recall.

3.4 Examples

In this section, three examples of computed evaluation scores are discussed.

Because there is only one class associated with each instance, it is always the case that $\sum_{\lambda=1}^{q} fp_\lambda = \sum_{\lambda=1}^{q} fn_\lambda$, meaning that the micro-averaged (version 1) precision and micro-averaged recall are always the same[3]:

$$\begin{aligned} precision_{micro} = recall_{micro} &= \frac{\sum_{\lambda=1}^{q} tp_\lambda}{\sum_{\lambda=1}^{q} tp_\lambda + \sum_{\lambda=1}^{q} fp_\lambda} \\ &= \frac{\sum_{\lambda=1}^{q} tp_\lambda}{\sum_{\lambda=1}^{q} tp_\lambda + \sum_{\lambda=1}^{q} fn_\lambda} \end{aligned} \tag{15}$$

3.4.1 Different frequency in train and test

Table 3 shows an example for which frequencies of the instances differ in training and test data: $f_\lambda^{test} \neq f_\lambda^{train}$. The values of the true positives (tp), false positives (fp), and false negatives (fn) come from the data, as well as the training frequencies f_λ^{train}. All other figures can be computed from these givens.

As can be seen, the two versions of the micro-averaged precision and recall differ.

[3] q is the number of labels in the training data. The equation would no longer hold if the number labels in the test data would be used and there would be a label in the training data that is not in the gold standard of the test data.

label λ	tp	fp	fn	precision	recall	f_λ^{test}	f_λ^{train}
c_1	3	2	7	0.6	0.3	0.25	0.1
c_2	1	7	9	0.125	0.1	0.25	0.2
c_3	6	6	8	0.5	0.429	0.35	0.3
c_4	6	9	0	0.4	1	0.15	0.4
total	16	24	24			1	1
macro-averaged				0.406	0.457		
micro-averaged				0.4	0.4		
lfb-micro-averaged				0.395	0.579		

Table 3: Different frequencies.

3.4.2 Same frequency in train and test: $f_\lambda^{test} = f_\lambda^{train}$

label λ	tp	fp	fn	precision	recall	f_λ^{test}	f_λ^{train}
c_1	3	2	7	0.6	0.3	0.25	0.25
c_2	1	7	9	0.125	0.1	0.25	0.25
c_3	6	6	8	0.5	0.429	0.35	0.35
c_4	6	9	0	0.4	1	0.15	0.15
total	16	24	24			1	1
macro-averaged				0.406	0.457		
micro-averaged				0.4	0.4		
lfb-micro-averaged				0.416	0.4		

Table 4: Same frequency in test and training for each label.

The data in Table 4 is build up in the same way as Table 3, but it comes from a different experiment. In this experiment, the frequency of a label in the test data is the same as in the training data. As can be seen, this leads to the fact that the two versions of the micro-averaged recall are the same. The micro-averaged precisions are still different.

3.4.3 Same frequency in train and test and $fp_\lambda = fn_\lambda$

The data in Table 4 is build up in the same way as the previous tables, but for this experiment, not only are the test and training frequencies the same, also, for each label, the number of false positives equals the number of false negatives. As can be seen, this leads to similar micro-averaged scores.

label λ	tp	fp	fn	precision	recall	f_λ^{test}	f_λ^{train}
c_1	6	4	4	0.6	0.6	0.25	0.25
c_2	6	4	4	0.6	0.6	0.25	0.25
c_3	6	6	6	0.5	0.5	0.3	0.3
c_4	6	2	2	0.75	0.75	0.2	0.2
total	24	16	16			1	1
macro-averaged				0.613	0.613		
micro-averaged				0.6	0.6		
lfb-micro-averaged				0.6	0.6		

Table 5: Same frequency in test and training and for each label $tp = fn$.

4 Which version to choose?

[[INSUFFICIENT: TO BE UPDATED]]

As Sebastiani (2002) argues it is hard to choose between micro- and macro-averaged scores:[4]

There is no complete agreement among authors on which is better. Some believe that "microaveraged performance is somewhat misleading (. . .) because more frequent topics are weighted heavier in the average" [Wiener et al. 1995, page 327] and thus favour macroaveraging. Others (actually, the majority of researchers) believe that topics should indeed count proportionally to their frequence, and thus lean towards microaveraging.

For single-label problems, choosing for micro-averaged scores would mean choosing for accuracy and losing the precision/recall difference.

In a following section, we will add another argument that can be used to choose between macro- and micro-averaging: the influence of replacing an error with another error.

4.1 Macro-averaged

For the macro-averaged scores, there are two possible computations: using the set of labels in the training data or using the set of labels in the test data. If both sets are equal, this is not an issue. But, when running an evaluation

[4]It should be noted that in the quote micro-averaging, means version 1.

script, while supplying it only with the labeled test file, one must be aware that there may be labels that are not present in the test file.

Comparison of two test files, containing a different set of predicted[5] class labels, is impossible if only the test files are given to the scoring algorithm. For example, consider two test files for which the sum of the F1-scores of all labels is 1.8 in both cases. In test file 1, there are three different class labels present. But in test file 2, there are only two class labels present. This would lead to a macro-averaged F1-score for of 0.6 for test file 1 and a macro-averaged score of 0.9 for test file 2. This difference is counter-intuitive, both files should have the same macro-averaged F1-score. The macro-averaged F1-score should not depend on the selection of class labels.

To make the comparison possible, one should make sure that all labels are available in each test file or that the training file is supplied to the evaluation script. For this reason, it is always better to use the set of labels from the training file when scoring test files.

5 Multi-label instances

In the previous sections, we assumed that there is only one class label per instance. Tsoumakas et al. (2010) descibe how to compute evaluation scores when an instance can have more equivalent class labels. Hereafter, the method of calculation used by the `confusionmatrix.py` script is described. This method is similar to the label-based method of Tsoumakas et al. (2010).

5.1 Counting: tp, fp, fn, tn

Each instance can have 0 or more labels.

[5] Or gold class labels, it depends from where the evaluation script extracts its set of possible class labels.

true positive (tp) :	+1 for a given label, if that label occurs in the gold-standard multi-label and the predicted multi-label,
false positive (fp) :	+1 for a given label, if that label occurs in the predicted multi-label and not in gold-standard multi-label,
false negative (fn) :	+1 for a given label, if that label occurs in the gold-standard multi-label and not in the predicted multi-label,
true negative (tn) :	the total number of occurrences that are not a given label minus the fps of that label.

5.2 Confusion matrix

A different number of labels in the gold-standard and predicted multi-labels leads to split counts in the confusion matrix. A split count means that a cell in the matrix has a different value, depending if you want to add all values of a row or a column. A split value of 1/3 means that for getting the row total (= number of occurrences in gold) you need to take value 3. For getting the column total (=number of occurrences in predicted), you need to take value 1.

5.3 Empty predictions

It may be allowed to have an instance without any label in the gold standard or to have no labels in the predicted multi-label. This empty label, [ ] or NONE, may be treated in two ways. It can be treated as any other label when computing the macro-averaged scores or it can be discarded.

For example, take a set of three single labels that can be predicted [A B C]. When computing the macro-average F-score, one may compute the average by division by 3. Not counting the empty label as a label. Or one may divide by 4 when the empty label is treated as a label. Thus, the actual label set becomes [A B C NONE]. But, the NONE label can never co-occur with another label.

5.4 Double predictions

Note that the `confusionmatrix.py` script does not simplify the labels. Thus,

GOLD LABELS : [A]
PREDICTED LABELS : [A A A]

leads to one tp and two fps. If you want this to lead to a single tp, you need to make sure there are no doubles in the predicted class labels.

5.5 Example

Consider the following multi-label predictions (two instances):

GOLD LABELS : [A] and [A]
PREDICTED LABELS : [B C] and [A]

The counts are given in Table 6 and the associated confusion matrix is depicted in Table 7.

label	tp	fp	fn	tn	precision	recall
A	1	0	1	0	1	0.5
B	0	1	0	1	0	0
C	0	1	0	1	0	0
total	1	2	1	2		

Table 6: Counts for the example multi-label predictions.

In the confusion matrix, wrong predictions may lead to split counts. For example, consider the gold-standard multi-label [A] and the predicted multi-label [B C] of size n=2.

This leads to the increase of the cell {A,B}[6] with 1/n=0.5 and also of the cell {A,C} with 1/n=0.5 for the row-totals. Leading to a row-total of 1, namely the number of times A occurs in the gold.

For the column totals, both cells are increased with 1. Leading to a column total of 1 for B and for C, because both occur once in the prediction.

6 The costs of changing predictions

6.1 Label-frequency-based micro-averaged

During the comparison of the predictions of two systems that are trained and tested on the same training and test set, there are 2 movements of the true

[6]The cell is identified as {gold label, predicted label}

	A	B	C	$\sum$
A	1	1\0.5	1\0.5	2
B	0	0	0	0
C	0	0	0	0
$\sum$	1	1	1	3\2

Table 7: Confusion matrix for the example multi-label predictions. The gold labels are shown in the column, the predicted labels in the row.

positive, false positive and false negative counts possible. Namely:

1. false positive → false positive: both systems make an incorrect prediction, but they choose a different label
2. true positive → false positive + false negative: one system makes a correct prediction and the other makes an error

Movement 1 only has an influence on the precision scores, because neither the true positives or the false negatives are involved. Movement 2 has an influence on the precision and the recall.

For movement 2, it is clear what the best direction is: always producing as much true positives as possible. The label for which true positives are created does not matter.[7] For movement 1, the choice is more complicated. should you prefer a false positive for label A or for label B? And should it matter which choice you make? In this section, more details are given of the influence of these movements in the lfb-micro-averaged and macro-averaged scores.

6.1.1 Precision

When calculating a lfb-micro-averaged precision score, the value depends on the label that was falsely predicted. For example, if you have an instance labeled with class A, then, in general, it has an influence whether the erroneous prediction is class B or class C.

The cost of the switch from class B to class C can be calculated:

[7]The two movements are not independent. There is a movement 1 included in movement 2. This means that, although the preferred direction is fixed, the false positive that disappears does have an influence. Nevertheless, it is hard to attach a prediction strategy to this observation. This is the reason why this section focuses on movement 1.

$$factor_\lambda(\delta) = -\frac{\delta tp_\lambda f_\lambda}{(tp_\lambda + fp_\lambda)(tp_\lambda + fp_\lambda + \delta)} \tag{16}$$

With:

δ : the number of false positives that move.
If the label will be giving false positives, δ is negative.
If the label will be receiving false positives, δ is positive,
tp_λ : the true positives of class λ before the label change,
fp_λ : the false positives of class λ before the label change,
f_λ : the frequency of class λ in the training data.

Thus, $factor_B(-1)$ is the marginal precision cost when a false positive of class B becomes a false positive of another label and $factor_C(1)$ is the marginal cost when a false positive of another label becomes a false positive for class C.

The difference in lfb-micro-averaged precision ($\Delta p_{B \underset{\rightarrow}{\delta} C}$) when δ false positives move from label B to label C is given by:

$$\Delta p_{B \underset{\rightarrow}{\delta} C}(\delta) = factor_B(\delta) + factor_C(\delta) \tag{17}$$

If $\Delta p_{B \underset{\rightarrow}{\delta} C}$ is positive, the switch leads to a precision increase.

It is possible to combine multiple false positive switches. Consider, switching δ_1 false positives going from class B to class C and δ_2 false positives going from class C to class D

$$\Delta p_{B \underset{\rightarrow}{\delta_1} C; C \underset{\rightarrow}{\delta_2} D} = factor_B(\delta_1) + factor_C(\delta_1 + \delta_2) + factor_D(\delta_2) \tag{18}$$

The knowledge of Δp may direct the choice of a class label in case of doubt. Indeed, the best strategy is often to always predict the same label in case of doubt. This will make the most $factor_\lambda$'s become positive and only one becomes negative. The choice of the label that should be predicted is more compicated, because of the interplay of the different factors. From tentative experiments, it appears to be that it is best to predict the label with the lowest frequency f_λ that occurs the most frequent in the test set and for which you think you are good at predicting the correct label. The aim is to select the label with the highest number of true positives and the lowest f_λ.

The influence of f_λ is more important than the influence of the true positives. So, choosing for the label with the lowest f_λ is a good strategy.

It is clear that the factor determining precision increase/decrease is rather complicated. Nevertheless, some approximative meaning can be attached to the behavior:

- The highest score is obtained when all false positives are grouped onto one of the least frequent classes (low f_λ).
- The lowest score is obtained when all false positives are spread almost evenly, but in such a way that the most frequent classes get a little bit more than the less frequent classes.

The question is what the advantage is of this behavior of *boosting* the precision. It will likely lead to higher lfb-micro-averaged precision scores, but are the results truly better? For a human, the amount of errors will be the same. Only in selected experiments, when the least frequent label is a good baseline according to the human interpretation, the precision bias may seem useful to capture. When the bias is not preferred, it may be a better option to report the first version of the micro-averaged precision or the precision per label.

Note that shifting false negatives to true positives per label also has an influence on precision. This influence is not explicitly calculated here, because it is harder to attach a prediction strategy to it.

6.1.2 Recall

For precision, the influence of shifting counts is rather complicated because two labels are involved: the giving and the receiving label. For recall, only one label is involveld. Indeed, it is always the case for each label λ that $tp_\lambda + fn_\lambda = c_\lambda = C^{te}$. Therefore, when a label receives an extra true positive, it loses a false negative. True positives and false positives are *communicating vessels*.

Similar to precision, we can compute the change for the lfb-micro-averaged recall when a label λ gets predicted differently:

$$\Delta r_{\underset{\rightarrow}{\delta}\lambda} = \frac{\delta f_\lambda}{tp_\lambda + fn_\lambda} \tag{19}$$

With:

δ : the number of true positives that move.
If the label will be giving true positives, δ is negative.
If the label will be receiving true positives, δ is positive,
tp_λ : the true positives of class λ before the label change,
fn_λ : the false negatives of class λ before the label change,
f_λ : the frequency of class λ in the training data.

If δ is positive, Δr is positive, meaning that the lfb-micro-averaged recall increases.

When we express the label frequency in the test set as: $f_{t,\lambda} = \frac{tp_\lambda + fn_\lambda}{N}$ with N the total number of instances. The equation becomes:

$$\Delta r_{\underset{\rightarrow}{\delta}\lambda} = \frac{\delta}{N} \frac{f_\lambda}{f_{t,\lambda}} \tag{20}$$

It is possible to combine, shifts for different labels:

$$\Delta r_{\underset{\rightarrow}{\delta_1} A; \underset{\rightarrow}{\delta_2} B;} = \Delta r_{\underset{\rightarrow}{\delta_1} A} + \Delta r_{\underset{\rightarrow}{\delta_2} B} \tag{21}$$

From the formula of Δr, it can be deduced that it is not the case that the most frequent class label has the most influence on lfb-micro-averaged recall. When the label frequency f_λ comes from the training corpus, the label that occurs more frequent in the training corpus than in the test corpus will have the most influence. It is as if the equation makes up for labels that are underrepresented in the test corpus. If an error is made on these labels, they will count heavier because they are more important in the training corpus. If the relative frequencies are the same in test and training set, as is the ideal situation, this correction mechanisms is inactive.

Sometimes the training corpus is not available and f_λ is taken from the test corpus. When the label frequency f_λ comes from the test corpus, Δr reduces to $\frac{\delta}{N}$ with N the number of instances. Thus, each label has a similar influence on the lfb-micro-averaged recall.

Note that shifting between true positives and false negatives for a label influences the false positives of another label. This influence is not calculated here.

6.2 Macro-averaged

We can do the same for macro-averaging. For precision, the factor becomes:

$$factor_\lambda(\delta) = -\frac{\delta tp_\lambda}{C(tp_\lambda + fp_\lambda)(tp_\lambda + fp_\lambda + \delta)} \tag{22}$$

With C the number of different classes.

As can be seen, giving false positives (negative δ) leads to precision gain. In most cases, moving all false positives onto one label so that only one label has to receive is the best strategy. Of course, during labeling, you don't know which are false positives but a good strategy can be to predict always the same class in case of doubt.

Which label to predict is harder to determine. You need to choose the label with the lowest factor while the other factors are the highest, but it is hard to intuitively grasp the complex interplay of δ, the false positive, and the true positive counts. From tentative experiments, it appears to be the case that the best label to predict is the label that occurs frequent in the test set and for which you think you are good at predicting it. The aim is to predict the label with the highest number of true positives in case of doubt. This conclusion is in concordance with the choice of the label for the lfb-micro-averaged precision, but for macro-averaged precision the f_λ's do not interfere.

For recall, the cost becomes:

$$\Delta r_{\underset{\rightarrow}{\delta}\lambda} = \frac{\delta}{C(tp_\lambda + fn_\lambda)} \tag{23}$$

Thus, the most frequent label in the test corpus, has the least influence. This behavior mitigates the frequency differences in the test corpus and it can be seen as the prove for the statement of Manning et al. (2008) that *to get a sense of effectiveness on small classes, you should compute macro-averaged results.*

6.2.1 Example

(To compute the changed predictions the script at
http://www.clips.ua.ac.be/∼vincent/software.html#finagle is used.)

	Precision	Recall	F1-score	Accuracy
NORMAL	78.11%	71.77%	73.05%	92.60%
CHANGED	78.87%	71.02%	73.16%	91.58%
UPPER	90.44%	71.83%	76.33%	93.32%

Table 8: Macro-averaged scores for a part-of-speech tagging experiment.

We have a part-of-speech system, based on the BNC (2001). Running an experiment gave the results of the first row (NORMAL) in Table 8.

We used a non-optimized memory-based kNN-based classifier.[8] The distance to the nearest neighbor is taken as a confidence score. A higher distance indicates a lower confidence for the prediction. Figure 1 shows that a lower confidence (higher distance) is indeed leading to more classification errors and thus a lower accuracy. This means this confidence score makes sense.

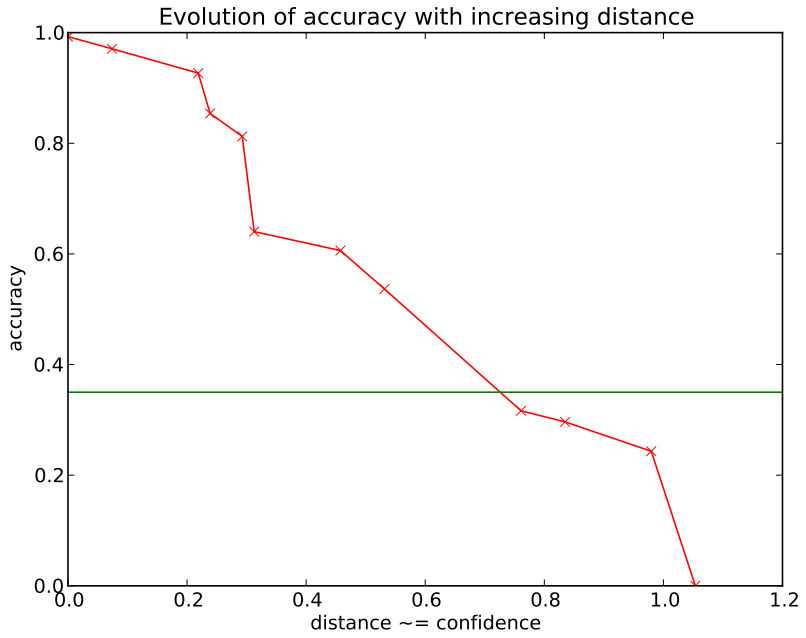


Figure 1: The accuracy per distance for a part-of-speech experiment.

With a script, all predictions of the part-of-speech tagging system that are at such a distance that the expected accuracy drops below 35% are changed. We are relatively confident these predictions will be wrong and, although there is no way to find the correct label, Formula (22) says that we best change

[8]TiMBL http://ilk.uvt.nl/timbl with options +v di+db.

these predictions to the most frequent tag. After this change the scores become those of the second row CHANGED in Table 8. Our confidence score was sufficiently helpful to identify enough wrong predictions to increase the macro-averaged precision (and the F1-score). Losing a few correct predictions leads to a lower recall (and accuracy). If the confidence score would be better at identifying incorrect predictions, the performance could be *boosted* more.

The third row (UPPER) of Table 8 are the results when all incorrect predictions would have been identified and altered to the most frequent tag. As can be seen, the precision is a lot higher. Some instances should have been labeled with the most frequent tag, these instance now receive the correct tag, leading to a slight recall increase.

This example shows that macro-averaged precision can be increased without having to make more correct predictions. The hard part is identifying incorrect predictions. And because identifying incorrect predictions is not 100% fail-safe, boosting precision by tinkering with the predicted labels is not often helpful. Furthermore, the confidence score that we proposed is not equally informative for every classification problem.

6.3 Micro-averaged, version 1

For the first version of the micro-averaged scores, shifting false positives has no influence on the averaged precision.

The recall difference of shifting δ true positives and false negatives is:

$$\Delta r(\delta) = \frac{\delta}{N} \tag{24}$$

With N the number of instances.

The precision difference of shifting δ true positives and false negatives is:

$$\Delta p(\delta) = \frac{\delta tp}{(tp + fp)(tp + fp + \delta)} \tag{25}$$

With tp and fp the counts summarized over all labels.

All these differences are independent for the label for which counts are shifted. This means that a label for which there are more instances in the test set, has more potential to weigh on the averaged scores as a group.

7 Evaluation in cross-validation

Forman & Scholz (2010) suggest how the evaluation in a cross-validation set-up should be done. Although Forman & Scholz (2010) inform on more averaging methods, we focus on two: micro-average style ($F_{tp,fp}$) and macro-average style (F_{avg}).

Consider a k-fold cross-validation setup. For each of the k folds, there is a system output. How to summarize all outputs into 1 evaluation score?

micro-average style: This averaging method means that all true positives, false positives, false negatives, and true negatives are summed over all folds. With these counts, an F-score, $F_{tp,fp}$ can be computed. It is possible to collect all tp's, fp's and fn's per class, such that we can compute the macro-averaged F-score from Section 3.1 in a micro-average way.

macro-average style: An F-score is computed for each fold. Next, the average of all these F-scores is computed.

As Forman & Scholz (2010) stress the unbiased way to combine output of cross-validation is in a micro-averaged manner.

7.1 Deviation between $F_{tp,fp}$ and F_{avg}

Assume that we carry out a regular test-train split experiment with two classes, class A and class B. The F1-score of class A can be computed as:

$$F1 = \frac{2tp_A}{N + tp_A - tp_B} \tag{26}$$

with

tp_c = true positives for class c

N = the total number of instances in test

Assume that we now carry out k-fold cross-validation using only the test set. In this case, our full test set is split into k test sets. Assume that we know that each newly trained machine learner is able to obtain a recall (=accuracy) of 80% on the positive class and a recall of 97.7% on the negative class.

Let's now consider the micro-averaged and macro-averaged style of averaging.

For the micro-averaged style, the formula of the F1-score of class A becomes:

$$F1_{tp,fp} = \frac{2\sum_k^K tp_A^k}{N + \sum_k^K tp_A^k - \sum_k^K tp_B^k} \tag{27}$$

with

$$\begin{aligned} \mathrm{tp}_c^i &= \text{true positives for class c in fold i} \\ K &= \text{the number of folds} \\ N &= \text{the total number of instances in test} \end{aligned} \tag{28}$$

For brevity, $\sum_k^K tp_A^k - \sum_k^K tp_B^k$ is substituted with C, the difference between the true positive counts of the two classes.

The macro-averagd style version is:

$$F_{avg} = \frac{1}{K}\sum_k^K \frac{2tp_A^k}{n^k + tp_A^k - tp_N^k} \tag{29}$$

with

$$\begin{aligned} \mathrm{tp}_c^i &= \text{true positives for class c in fold i} \\ K &= \text{the number of folds} \\ n^i &= \text{the total number of instances in the test set of fold i} \end{aligned} \tag{30}$$

Again for brevity, we can substitute $tp_A^k - tp_N^k$ with c^k and $\sum_k^K c^k = C$.

We can now compute the situation in which both are the same:

$$\begin{aligned} \frac{2\sum_k^K tp_A^k}{N + C} &= \frac{1}{K}\sum_k^K \frac{2tp_A^k}{n^k + c^k} \\ \sum_k^K tp_A^k &= \sum_k^K \frac{N + C}{K(n^k + c^k)}\ tp_A^k \end{aligned}$$

We call $\frac{N+C}{K(n^k+c^k)}$ factor δ. And this δ varies for each fold. Although there may be more solutions that make that $F1_{avg}$ equals $F1_{tp,fp}$, a straightforward

solution is when each δ equals 1. If a δ diverges from 1, most probably $F1_{avg}$ diverges from $F1_{tp,fp}$.

We can now discuss which parameters influence the difference between $F1_{avg}$ and $F1_{tp,fp}$. When does a δ differ from 1:

the number of instances per fold Ideally, each fold contains the same number of instances, meaning that $Kn^k = N$. But sometimes the total number of instances cannot be divided by the number of folds without a remainder. When this remaining instances are also distributed over the folds, n^k may differ from fold to fold.

the difference in true positives between the two classes Ideally, when instances are randomly distributed over the folds, the difference between the true positives is the same for each fold and $Kc^k = C$. But when the corpora are small or when the proportions of the classes are skewed, it may be impossible to distribute the true positives of each class evenly over all folds.

the number of folds When K becomes larger, the differences for n^k and, more important, the difference for c^k become more outspoken.

To summarize, $F1_{avg}$ equals $F1_{tp,fp}$ when instances can be evenly distributed over the folds of equal size. Any deviation from an exactly even distribution will cause $F1_{avg}$ to diverge from $F1_{tp,fp}$. This effect will be more pronounced when the number of folds increases.[9]

If $F1_{avg}$ and $F1_{tp,fp}$ can be different, which version to choose? Figure 2 shows the evolution of the F1-scores. When keeping in mind that less instances probably means that c^k will differ in each fold and that a lower proportion[10] also means that c^k will differ more easily, the effect of varying δ factors can be seen in this figure.

When looking at Figure 2, no matter what K or N is chosen, the evolution of $F1_{tp,fp}$ remains the same.[11] For this reason, micro-averaging style is the

[9] Similar reasonings can be derived for recall and precision.

[10] At the left of the figures.

[11] It is normal that F1-score varies when the accuracy for class A and class B remains the same and the proportion of class A in the corpus changes. In the experiments of Figure 2, we keep the accuracies constant because this means that the machine learner remains the same. In Forman & Scholz (2010), the precision and recall of class A are kept constant, but in order to maintain these scores constant, the accuracy of class B should change when the proportion of class A varies. This would mean that the machine learner should produce different results for class B depending on the proportion of class A in the test corpus. This is not a realistic situation.

preferred method to compute scores when cross-validating.

It is possible to compute the evolution of the macro-F1 score in function of p, the proportion of the positive class A in the corpus.

$$\begin{aligned} macroF1 &= \frac{tp_A}{2tp_A + fp_A + fn_A} + \frac{tp_B}{2tp_B + fp_B + fn_B} \\ &= \frac{r_A}{1 + r_A + (1 - r_B)p} + \frac{r_B p}{1 - r_A + (1 + r_B)p} \end{aligned} \tag{31}$$

with

$$\begin{aligned} n_c &= \text{number of instances of class c} = tp_c + fn_c \\ p &= \frac{n_B}{n_A} \\ r_c &= \text{recall of class c} = \frac{tp_c}{n_c} \\ fp_i &\text{ equals } fn_j \text{ when } i \neq j \end{aligned}$$

Equation 31 is plotted in Figure 2c (theoretical). It coincides almost exactly with the curve of $F_{tp,fp}$.[12] This is again an argument to choose for the micro-average style of averaging during cross validation.

When the number of instances in the test set is *large*, the difference between $F1_{tp,fp}$ and $F1_{avg}$ is small.

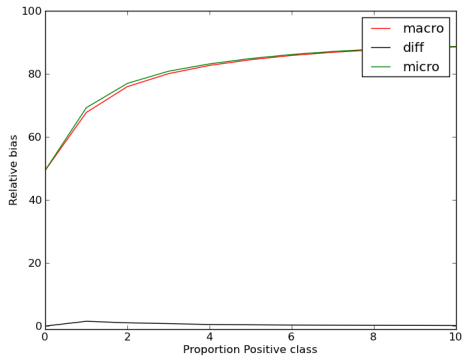
8 Statistical significance

[[TODO]]

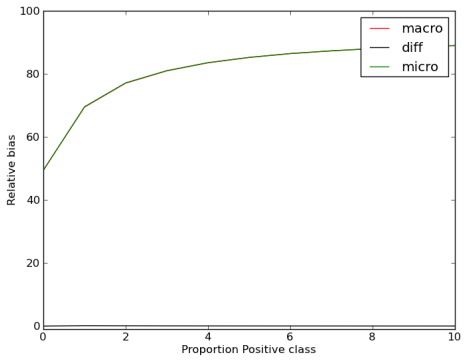
See http://www.clips.ua.ac.be/∼vincent/tutor/histogram.pdf

In Dutch and contains no references. For approximate randomization tests, at least: Noreen (1989), Yeh (2000) and Yang & Liu (1999)

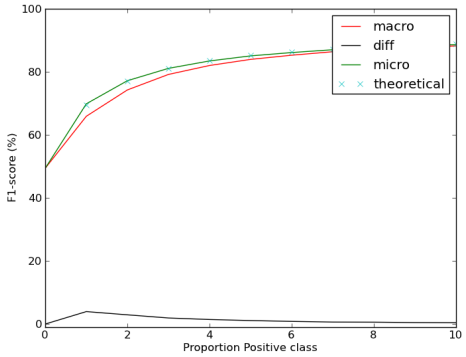
[12] Note that for the curve of Figure 2c, the substitution of $p = \frac{1-prop_A}{prop_A}$ has been used, since the x-axis shows the proportion of class A ($prop_A$), which is $\frac{n_A}{n_A+n_B}$.



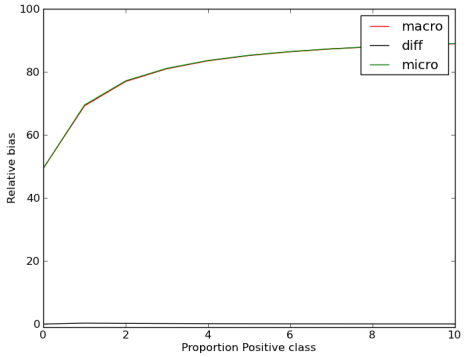
(a) 5 folds and 1000 instances



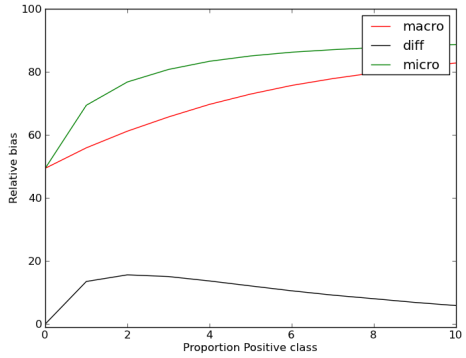
(b) 5 folds and 10,000 instances



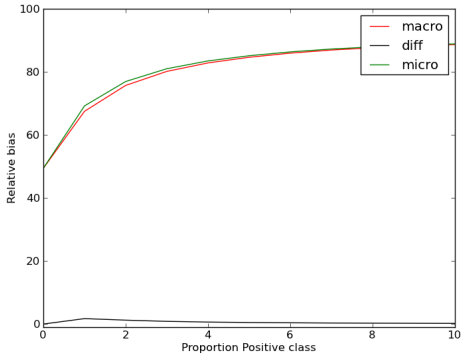
(c) 10 folds and 1000 instances



(d) 10 folds and 10,000 instances



(e) 50 folds and 1000 instances



(f) 50 folds and 10,000 instances

Figure 2: Evolution of the macro-averaged F1-score when the data contain propoprtions of class A ranging from 0% to 10%. The accuracy for class A is fixed at 80%, the accuracy of class B is fixed at 97.7%. TODO: remake the figures with F1-score for class A and rename Y-axis to Fscore (%)

9 Scripts

- **confusionmatrix.py** : This script can handle files with one or more class labels per instance. The label-frequency based micro-averaged scores are computed as in version 2. The computed micro-averaged scores are like version 1. It is possible to supply the training file to the script. The script provides more information about how the scores are calculated. (www.clips.ua.ac.be/∼vincent/software.html#confusionmatrix)
- **art.py** : Approximate randomization tests to assess the statistical significance of differences between scores. (www.clips.ua.ac.be/∼vincent/software.html#art)
- **scoreconverter** : An online demo to convert macro-averaged scores into true positives, etc. And many more combinations. (www.clips.ua.ac.be/cgi-bin/vincent/scoreconverter.html)

References

BNC (2001). The British National Corpus, version 2 (BNC world). Available at http://www.natcorp.ox.ac.uk (Last accessed: March 2013).

Daelemans, W., Zavrel, J., van der Sloot, K., & van den Bosch, A. (2010). Timbl: Tilburg memory-based learner, version 6.3. Tech. Rep. ILK 10-01, Tilburg University.

Forman, G., & Scholz, M. (2010). Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement. *SIGKDD Explorations Newsletter*, *12*(1), 49–57.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press.

Noreen, E. W. (1989). *Computer-intensive methods for testing hypotheses*. New York, NY, USA: John Wiley.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, *34*(1), 1–47.

Tsoumakas, G., Katakis, I., & Vlahavas, I. P. (2010). Mining multi-label data. In O. Maimon, & L. Rokach (Eds.) *Data Mining and Knowledge Discovery Handbook*, (pp. 667–685). Heidelberg, Germany: Springer-Verlag, 2nd ed.

van Rijsbergen, C. J. (1975). *Information Retrieval*. London, UK: Butterworths.

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods.

In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, (pp. 42–49). New York, NY, USA: ACM.

Yeh, A. (2000). More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International Conference on Computational Linguistics*, vol. 2, (pp. 947–953). Saarbrücken, Germany: Association for Computational Linguistics.