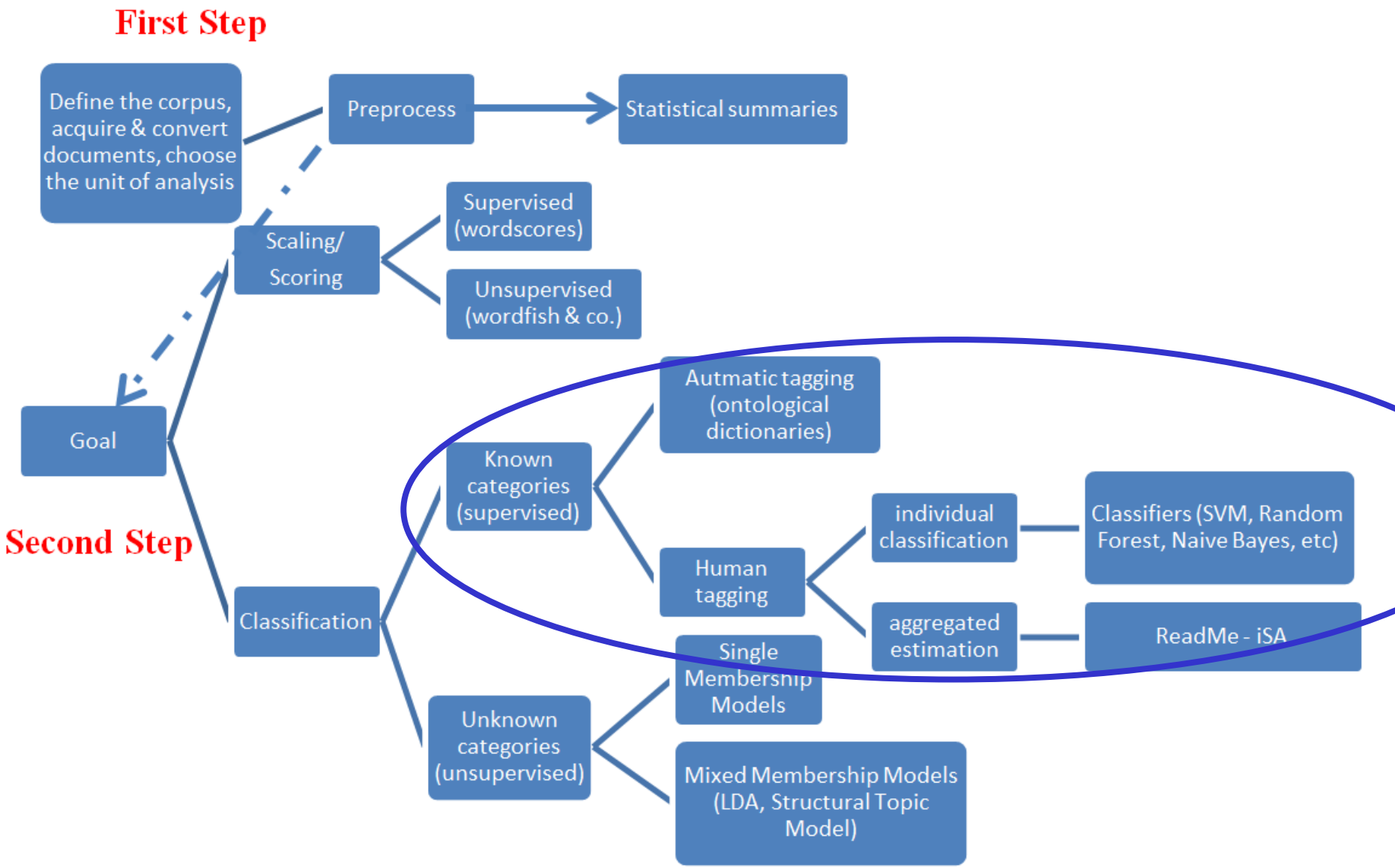
Big Data Analytics

Lecture 4/B
The importance of the training-set

UNIVERSITÄT LUZERN

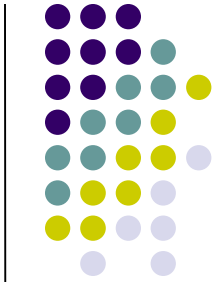
Our Course Map



Reference

- ✓ Grimmer, Justin, and Stewart, Brandon M. (2013). Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts. *Political Analysis*, 21(3): 267-297
- ✓ Curini, Luigi, and Robert Fahey (2020). Sentiment Analysis and Social Media. In Luigi Curini and Robert Franzese (eds.), *SAGE Handbook of Research Methods is Political Science & International Relations*, London, Sage, chapter 29
- ✓ Barberá, Pablo et al. (2020) Automated Text Classification of News Articles: A Practical Guide, *Political Analysis*, DOI: 10.1017/pan.2020.8

Constructing a training set



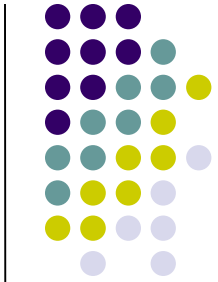
For supervised problems, the researcher is aiming to classify documents into a set of known or assumed categories based upon rules or information that can be learned from the training set

This requires **labels** in the training set from which to infer categories in the test set

The most important step in applying a supervised learning algorithm is therefore constructing a **reliable training set**, because no statistical model can repair a poorly constructed training set!

If the training set is **poorly constructed**, the supervised algorithms will simply replicate such poorly construction!

Constructing a training set



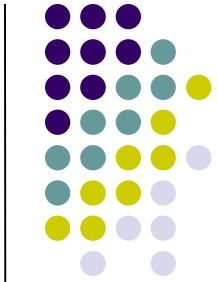
(1) creating and executing a coding scheme:

Best practice is to **iteratively develop coding schemes**

Initially, a **concise codebook** is written to guide coders, who then apply the codebook to an initial set of documents

When using the codebook, particularly at first, coders are likely to **identify ambiguities** in the coding scheme or overlooked categories

Constructing a training set



(1) creating and executing a coding scheme:

This subsequently leads to a **revision of the codebook**, which then needs to be applied to a new set of documents to ensure that the ambiguities have been sufficiently addressed

Only after coders apply the coding scheme to documents without noticing ambiguities is a “final” scheme ready to be applied to the data set

Constructing a training set

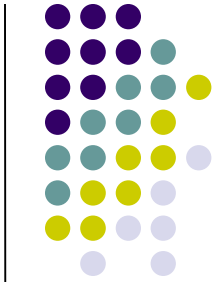
(1) creating and executing a coding scheme:

Finally, always define a number of exhaustive (and exclusive) categories

For example, you can also decide to include a "*Others*" category in your codebook, wherein classifying the texts that do not deal with any of the theoretically interesting categories you have identified for your research

ML algorithms must learn also that!

Constructing a training set



(2) **sampling documents:**

Basically all ML methods aiming at individual classification implicitly assume that the **training set is a random sample** from the population of documents to be coded

This is because Supervised learning methods **use the relationship** between the features in the training set to classify the remaining documents in the test set (out-of-sample predictions)

Constructing a training set

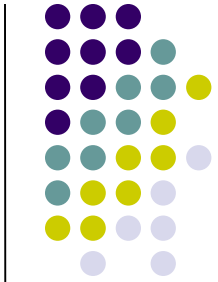
(2) **sampling documents:**

This presents particular difficulty when…

…**all the data are not available at the time of coding**: either because it will be produced in the future or because it has yet to be digitized

Per-se, this could be particularly problematic in dealing with any ***semantic change***, which is the difference in the meaning of language between the training and the test-set

Constructing a training set



(2) **sampling documents:**

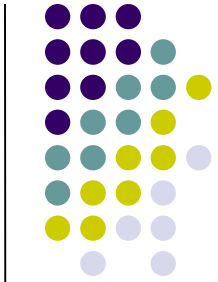
For example, we can have **emergent discourse**, where new words and phrases, or the meanings of existing words and phrases, appear in the unlabeled set but not the labeled set

…and **vanishing discourse**, where the words, phrases, and their meanings exist in the labeled set but not the unlabeled set

How to face this risk?

Keep updating the training-set (if your test-set is still to come...)!

Constructing a training set

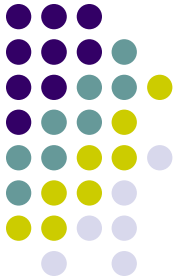


(2) **sampling documents:**

Moreover, Supervised methods need **enough information** to learn the relationship **between words and documents in each category of a coding scheme**

Hopkins and King (2010) offer **five hundred** as a rule of thumb with one hundred documents probably being enough

Constructing a training set

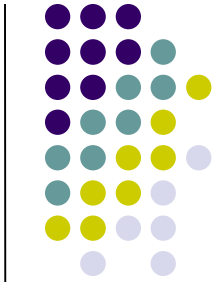


(2) **sampling documents:**

Still the number necessary will depend upon **the specific application of interest**. For example, as the number of categories in a coding scheme increases, the number of documents needed in the training set also increases

Moreover, if a category **occurs rarely** in the training set compared to the other categories, the "*curse of highly imbalanced training-set*" is waiting for you

Constructing a training set



(2) **sampling documents:**

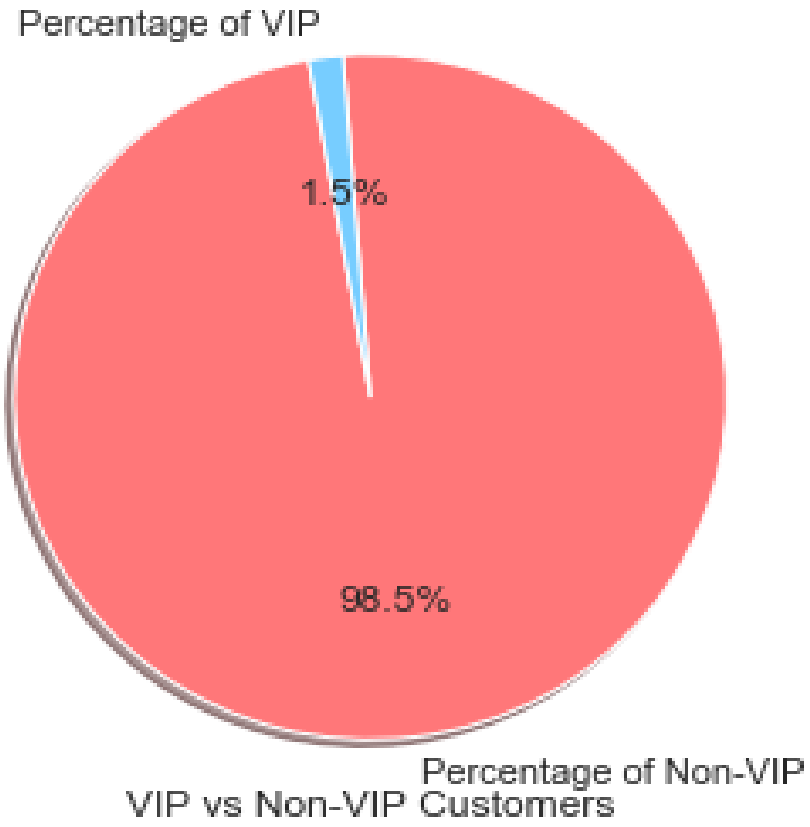
Given a highly imbalanced training-set, any ML risks in fact to perform badly because it is not trained on a sufficient amount of data representing the minority class(es) to "learn" about this category and its properties

This will affect negatively our out-of-sample prediction for this category (as well as, indirectly, for the majority class)

Constructing a training set

(2) **sampling documents:**

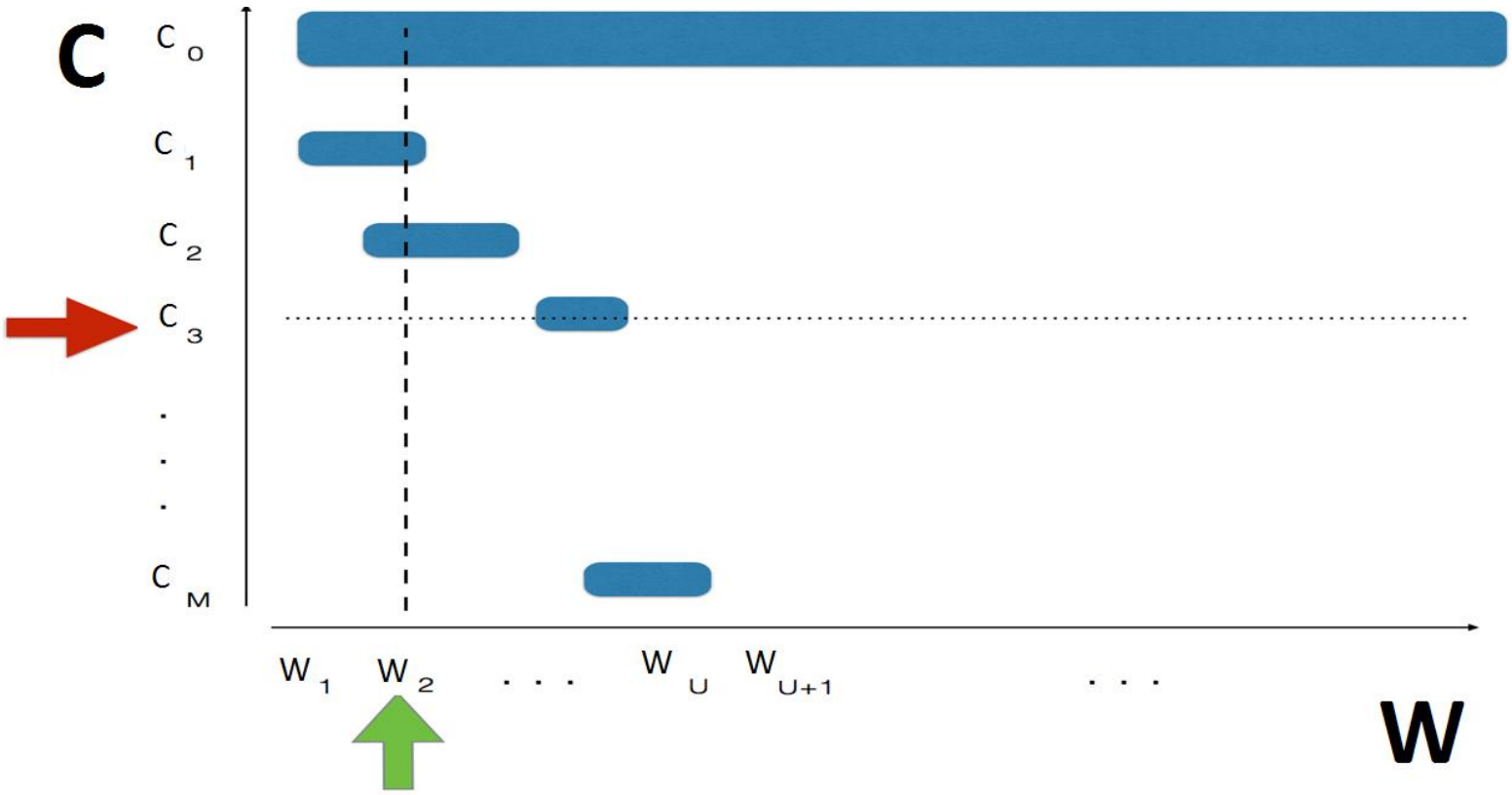
This is due to the influence that the larger majority class produces



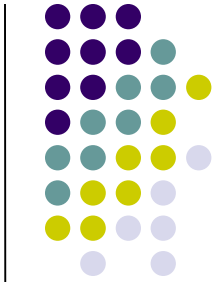
Constructing a training set

(2) **sampling documents:**

The existence of a category C_k extremely frequent in a training-set can in fact negatively affect $p(\boldsymbol{C}|\boldsymbol{W})$



Constructing a training set



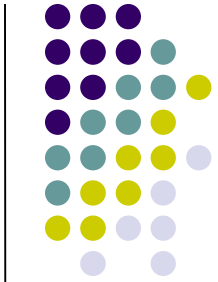
(2) **sampling documents:**

And so?

Best strategy: go back to your training-set and improve on it by collecting more texts for the minority categories to decrease the overall level of class imbalance

And if you cannot? As a second-best strategy, you can always try to **resample** the original training dataset

Constructing a training set



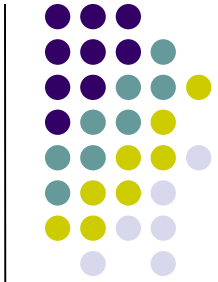
(2) **sampling documents:**

Resampling is done either by *oversampling* the minority class and/or *under-sampling* the majority class until the classes are approximately equally represented

Even though both approaches address the class imbalance problem, they also suffer some drawbacks

The random undersampling method can potentially remove certain important data points (and therefore information!), and random oversampling can lead to overfitting

Constructing a training set



(2) **sampling documents:**

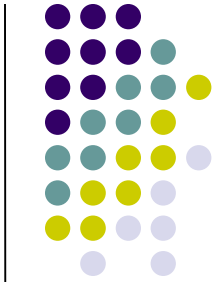
Other possibility: Synthetic data generation such as…

SMOTE: Synthetic Minority Over-sampling Technique has been designed to generate new samples that are coherent with the minor class distribution

The main idea is to consider the relationships that exist between samples and create new synthetic points along the segments connecting a group of neighbors

However always keep in mind that ML algorithm assumes that the training set is a **random sample** from the population of documents to be coded…

Constructing a training set



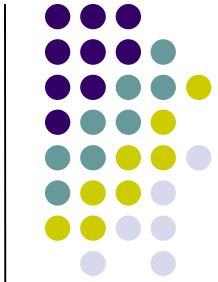
(3) **checking human-tagging reliability:**

Manually generating the initial set of labels can prove arduous and time-consuming, but is also fraught with concerns about consistency and accuracy

That is, while labeling training data often requires the use of human coders to sort texts into desired categories, human coding lacks consistency and reliability both within and across individuals, above and beyond the time and expense required to complete the task

Therefore always run an **inter-coder reliability text**!!!

Constructing a training set



(3) **checking human-tagging reliability:**

What is **inter-coder (or inter-rater) reliability**?

Intercoder reliability is the extent to which 2 different researchers agree on how to code the same content

It's often used in content analysis when one goal of the research is for the analysis to aim for consistency and validity

Intercoder reliability ensures that when you have multiple researchers coding a set of data, that they come to the same conclusions

Constructing a training set

One common statistics used is Cohen's kappa coefficient (κ)

k is a more robust measure than simple percent agreement calculation, as it takes into account the possibility of the agreement occurring by chance

How? K is estimated as $(p_o-p_e)/(1- p_e)$

where $p_o$is the relative observed agreement among raters (identical to accuracy), and p_e is the hypothetical probability of chance agreement, using the observed data to calculate the probabilities of each observer randomly seeing each category

If the raters are in complete agreement then k=1. If there is no agreement among the raters other than what would be expected by chance (as given by p_e), k=0. It is possible for the statistic to be negative, which implies that the agreement is worse than random

Usually a reasonable value for k is larger than .6 (but larger than .8 would be far better)

Constructing a training set

Let's see an example, with 2 coders, 2 categories, and 25 texts to code for each coders

	Coder B	
Coder A	Positive	Negative
Positive	20	5
Negative	10	15

Observed proportionate agreement (p_o): (20+15)/50=0.7

Constructing a training set

	Coder B	
Coder A	Positive	Negative
Positive	20	5
Negative	10	15

And the probability of a random agreement (p_e)?

- ✓ Coder A said “Positive" to 25 texts and “Negative" to 25 texts. Thus reader A said “Positive" 50% of the time.
- ✓ Reader B said “Positive" to 30 texts and "Negative" to 20 texts. Thus reader B said “Positive" 60% of the time

So the expected probability that both would say “Positive” at random is: 0.5*0.6=0.3

Similarly, the expected probability that both would say “Negative” at random is: 0.5*0.4=0.2

Overall random agreement probability is the probability that they agreed on either Positive or Negative, i.e. (p_e)=0.3+0.2=0.5

Constructing a training set

	Coder B	
Coder A	Positive	Negative
Positive	20	5
Negative	10	15

Applying the formula for Cohen's Kappa we get:

✓ $k=(p_o-p_e)/(1- p_e)=(0.7-0.5)/(1-0.5)=0.4$

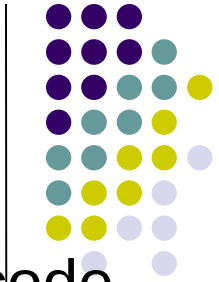
Constructing a training set

Given the likely existence of a budget constraint sometimes we will need to make a **choice between more coders per object and more texts coded**. What is better?

The literature shows that while increasing the number of coders for each document can improve the accuracy of the classifier…

…the informational gains from increasing the number of documents coded are greater than from increasing the number of codings of a given document

Constructing a training set



This does not obviate the need to have multiple coders code at least a subset of documents, namely to determine coder quality and to select the best set of coders to use for the task at hand

But once the better coders are identified, the optimal strategy is to proceed with one coder per document

R pakcages to install

install.packages("irr", repos='http://cran.us.r-project.org')