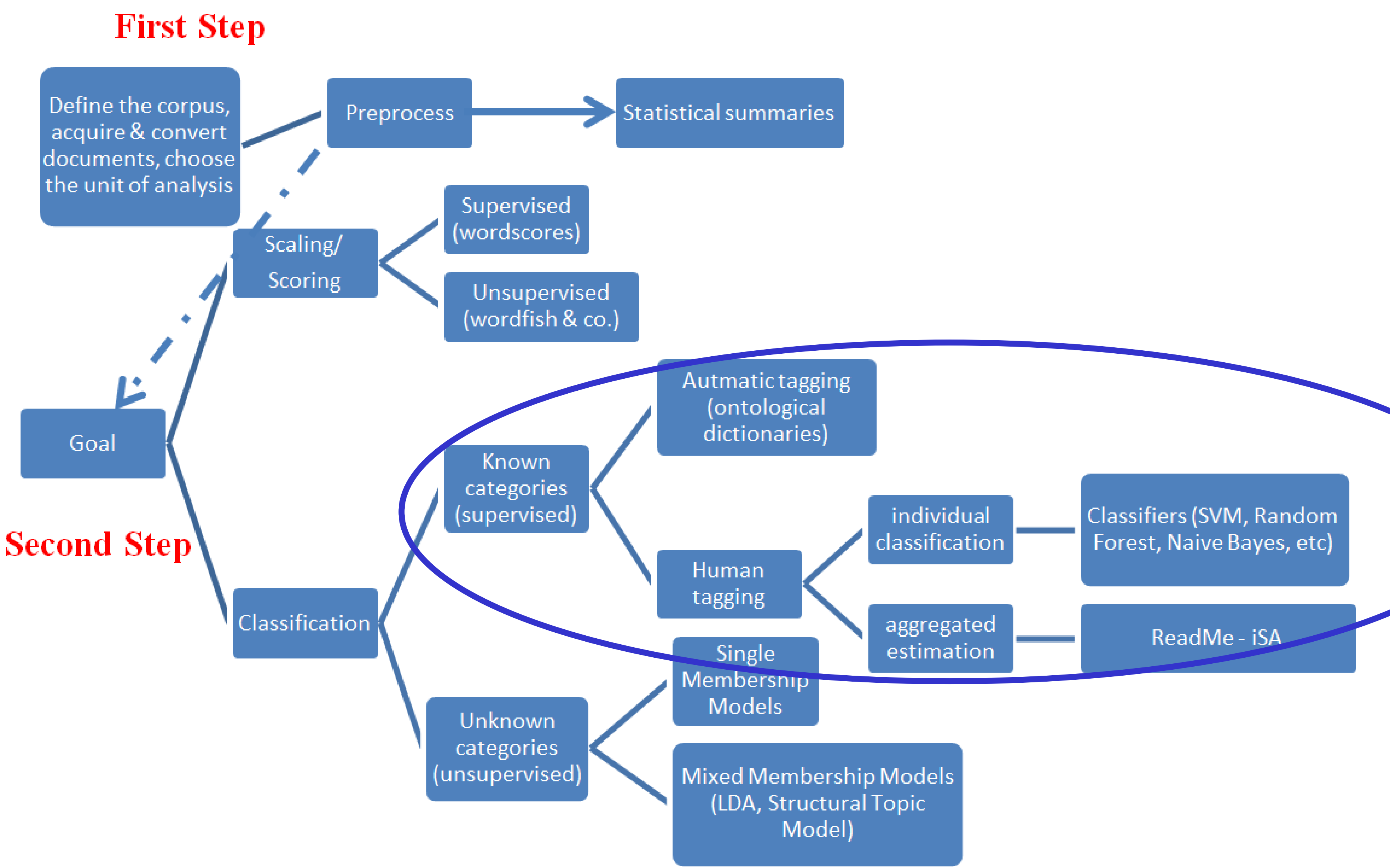
Big Data Analytics

Lecture 2/B

Dictionaries and Semisupervised classification methods

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
- ✓ Kohei Watanabe and Yuan Zhou (2020) Theory-Driven Analysis of Large Corpora: Semisupervised Topic Classification of the UN Speeches, *Social Science Computer Review*, DOI: 10.1177/0894439320907027
- ✓ Shusei Eshimay Kosuke Imaiz Tomoya Sasakix (2020). Keyword Assisted Topic Models, arXiv:2004.05964v1

Classification methods

Classifying Documents into Known Categories

Assigning texts to **some known categories** (rather that to categories *discovered ex-post* the analysis – as it happens with unsupervised classification methods) is the most common use of content analysis methods in political science

For example, researchers may ask if local news coverage is positive or negative, if legislation is about the environment or some other issue area, if international statements are belligerent or peaceful, etc.

In each instance, the goal is to infer to which - among a given set of pre-defined categories - each document must be assigned

Classification methods

There are **two broad groups of supervised classification methods** available according to the type of **tagging** (i.e., the assignation of a document to a given pre-defined category) **employed**:

We can have either:

1) human tagging - *supervised learning methods*
2) automatic tagging – *dictionaries*

Human tagging

- ✓ **Supervised learning methods** replicate the familiar manual coding task, *but* with a machine

First, human coders are used to classify a subset of documents into a predetermined categorization scheme

Second, this subset is used to train an automated method (i.e., a ML algorithm)

Finally, the automated method then classifies the remaining unread documents

- ✓ **Dictionaries** use on the contrary the **relative rate** at which key words appear in a text to **classify documents into categories**

Let's start discussed today about automatic tagging…

Dictionary methods

Suppose the goal is to measure the **tone** (also called the "**sentiment"**) in newspaper articles: whether articles convey information positively or negatively about a given topic

A **dictionary to measure sentiment** is a list of words that are either dichotomously classified as positive («good», «fantastic», etc.) or negative («bad», «horrible», etc.) or contain more continuous measures of their content

You can then use that dictionary to identify **the tone of a document**: either positive or negative according to the relative number of words in that document identified by the dictionary as positive or negative ones

Dictionary methods

Formally, within a given dictionary Z each word *m* (m=1,….M) will have an associated score s_m

For the simplest measures, s_m =-1 if the word is associated with a negative sentiment and s_m =+1 if associated with a positive sentiment

The analyst then applies some *decision rule*, such as summing over all the weighted feature values, to create a *single score* for the document

Dictionary methods

For example, if $N_i = \sum_{m=1}^{M} W_{im}$ are the words included in dictionary Z that are used in document *i*, then dictionary methods can use such list of words to measure the sentiment for any document t_i in the following way:

$$t_i = \sum_{m=1}^{M} \frac{s_m W_{im}}{N_i}$$

So for example, if document *i* presents the words «good», «fantastic» and «bad», then $t_i = 1/3$ or 0.333

Dictionary methods

Scholars often use t_i as an approximately continuous measure of document sentiment, that is, it allows us to sort documents as to which are more or less positive or negative relative to one other

t_i can also be used to classify documents into **sentiment categories** if a decision rule that identifies a cut point is assumed along with the dictionary method

Perhaps the simplest coding rule would assign all documents with $t_i > 0$ to a positive sentiment category and $t_i < 0$ to a negative sentiment

And if $t_i = 0$? Either neutral category or NC

Dictionary methods

Of course, the words included in the texts you are analyzing that are **not also included** in the dictionary, will not provide any additional information for your classification aim (we will discuss more about this point later on)

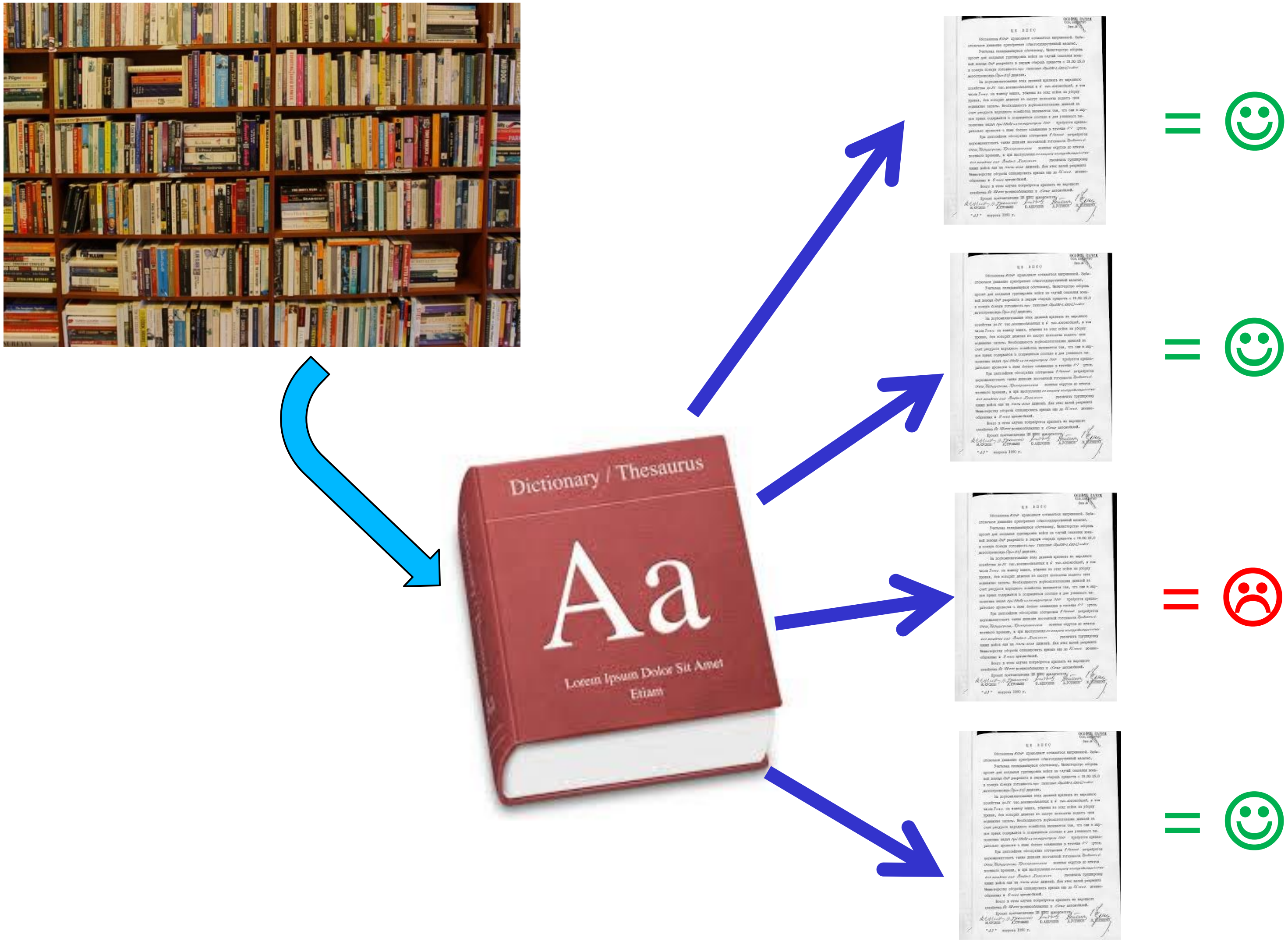
Dictionary methods

Sentiment analysis is just one type of analysis a dictionary method can perform

The general idea of dictionaries make them relatively easy and cheap to apply across a variety of problems: identify **words that separate categories** (for example *policy categories*) and measure **how often those words occur in texts**

For example, the Lexicoder Topic Dictionary (Albugh et al., 2013) contains 1,387 keywords under 28 topics (e.g., macroeconomics, civil rights, health care, agriculture) based on the Comparative Agenda Project’s coding scheme. If you are interested about it, just let me know!

Dictionary methods



Dictionary methods

Using a dictionary can therefore **minimize** the amount of labor needed to classify documents (no human involved in the tagging proces after all!)

This is very attractive! Once you have for example a sentiment dictionary, you can apply it to any corpus you have

But…beware of the challenges of using a dictionary!

Dictionary methods

For dictionary methods to work well, the scores attached to words must closely align with **how** the words are used in a particular context

If a dictionary is developed for a **specific application**, then this assumption should be easy to justify

But when dictionaries are **created in one substantive area and then applied to another**, serious errors can occur

Why that?

Dictionary methods

To build a “good” dictionary you need to be sure that all relevant terms are included in it (**no false negatives**, i.e., terms we should have included in the dictionary cause they are relevant given our research topic, but failed to do so)…

…and no irrelevant or wrong terms are (**no false positives**, i.e., terms we have included in the dictionary but should not have, being them irrelevant given our research topic)

But language do **change across topics**! And when this happens, false negatives and false positives proliferate!

For example, a word like `cancer` may have a positive connotation in a health-care company documents, but negative in many other contexts

Dictionary methods

Moreover, dictionary methods work pretty well when you study texts that use a **standardized language** (i.e., legal text!).

In other contexts, things become more complex…given that **language evolves continuously**: it is a social construction after all!

Dictionary methods

On the other side, it is almost impossible to code all possible semantic rules in a pre-defined dictionary (double meaning sentences, specific jargons, neologisms, irony)

Dictionary methods

Finally, **counting the number** of positive and negative terms in a sentence may lead to **paradoxical effects**

"*This movie has **good** premises. Looks like it has a **nice** plot, an **exceptional** cast, **first class** actors and Stallone gives his **best**. But it sucks*"

5 POSITIVE TERMS VS 1 NEGATIVE

Dictionary methods

Dictionaries, therefore, should be used with **substantial caution**, or at least coupled with **explicit validation**

The problem is that quite often measures from dictionaries are **rarely validated**

Rather, standard practice in using dictionaries is to assume the measures created from a dictionary are correct and then apply them to the problem

The consequence of **domain specificity and lack of validation** is that most analyses based on dictionaries are built on shaky foundations

Dictionary methods

If using dictionaries, choose therefore a dictionary **appropriate** to the task at hand, and **validate** the utility of the dictionary, for example by confirming that a sample of dictionary-generated scores of text in the corpus conform to human coding of the text for the measure of interest

Semisupervised classification

When using **topic models**, most researchers have the topics of interest in mind and possess (hopefully) a substantial amount of knowledge about them

After all, social scientists analyze textual data in order to empirically test hypotheses derived from substantive theories

Thus, researchers should find it natural to incorporate such **prior information** into topic models as keywords

In contrast, the standard topic models such are designed for the settings in which researchers wish to **explore** the contents of corpus, w/o any prior knowledge

Semisupervised classification

Using topic models (i.e., unsupervised methods) produces therefore results that sometimes are difficult to make sense of (and perhaps not the topics you would be interest in as a researcher…)

Using **dictionary models** (i.e., automatic supervised methods) could also be problematic for the reasons discussed. Moreover, quite often you do not have any dictionary available for the research topic you have in mind!

Semisupervised classification

Should we then move to a human supervised methods?

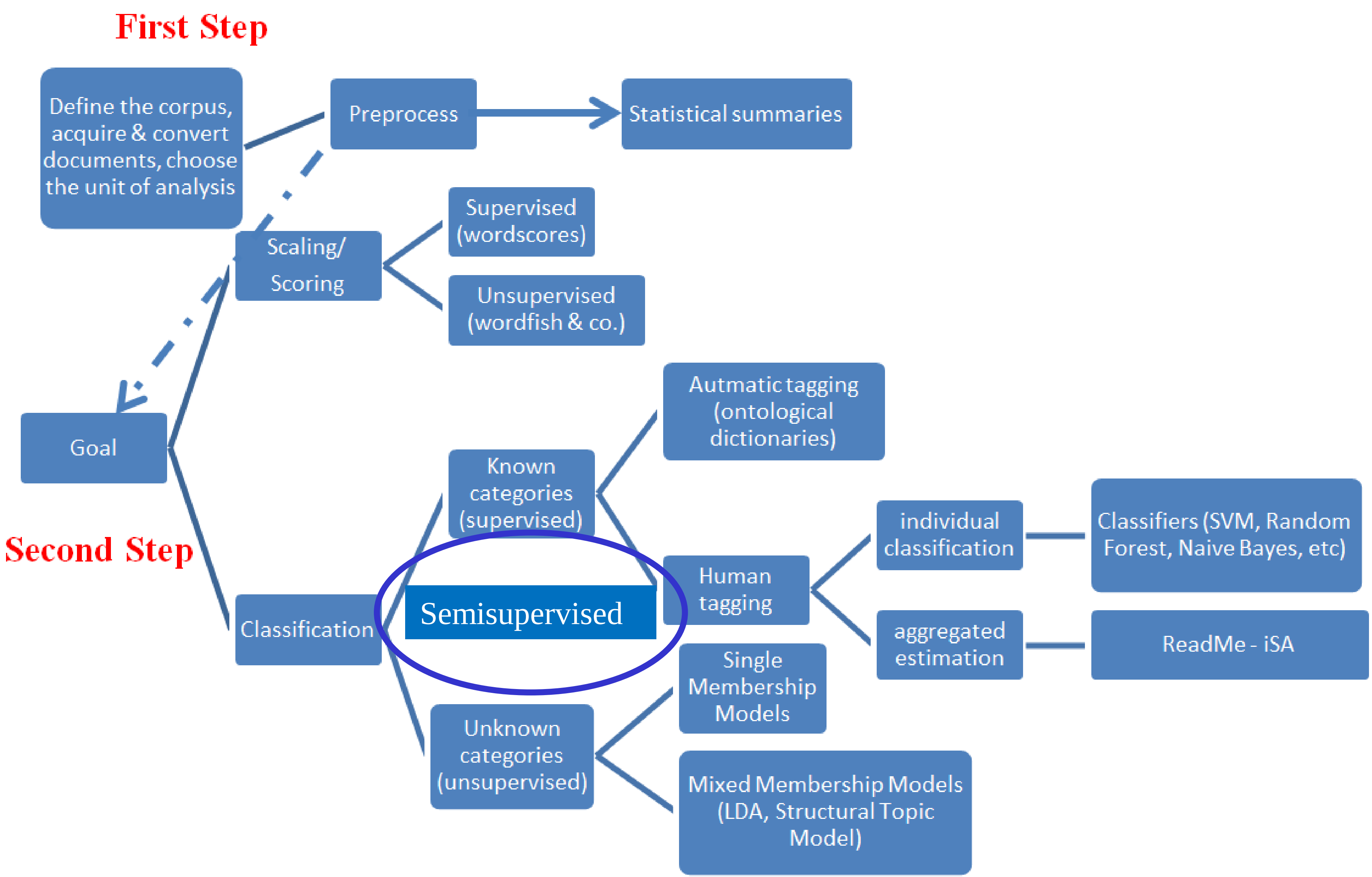
This could present several advantages (as we will discuss!), however it is quite time-consuming: you have to create a training-set from scratch after all (more on this later on)

And so?

Perhaps we could give a try to **semisupervised classification**

P.S. note that you can also have *semisupervised scaling algorithms*, such as Latent Semantic Scaling (implemented in R via the LSX package). If you are interested about it, just drop me an email!

Our Course Map



Semisupervised classification

How do they work? Let’s first discuss **Newsmap**

Suppose we want to classify the news in our corpus according to two class-labels: either if they discuss about Ukraine or Iraq

First thing you have to do is to identify some words that would help us to automatically create a dictionary

Let’s call these special words «**seed words**»

In our case, the seed words for countries such as Ukraine and Iraq could be {Ukraine, Ukrainian*, Kiev} and {Iraq, Iraqi*, Baghdad}

Seed words dictionary is the only manual input to the system, and serves as **semi-supervision**

Semisupervised classification

The researcher pre-defines a list of labels/categories in which she is interested (Ukraine and Iraq in our example)

The researcher identifies the **seed words** associated to each pre-defined categories (the only manual input to the system)

In the **training-stage** the algorithm takes advantage of these seed-words to automatically create a dictionary for each label

Then, in the **classification-stage** the algorithm classifies all the texts (also texts that do not include *any* of our seed-words) into one of the pre-defined categories

Semisupervised classification

The training-stage

Firstly, we search individual documents for keywords in the seed dictionary (simple keyword matching) and gives them class labels (countries)

For example, suppose that in the corpus we have the following text: “This is an article about *Ukraine*”

We are going to **automatically label** such document as “Ukraine” given that it includes at least one of the seed-word {Ukraine, Ukrainian*, Kiev}

Semisupervised classification

Secondly, we aggregate the frequency of words according to the class labels to create **contingency tables**, i.e., the labels are used to estimate the association between the labels and features

In the contingency table below, c_j is a country of interest (i.e., Ukraine) and $\bar{C}_j$ is all other countries; w_i is the word for which scores are calculated (say word “article”) and w'_i is all other words; *F* values are all raw frequency counts of words in respective classes

	c_j	$\bar{C}_j$
w_i	F_{11}	F_{01}
w'_i	F_{10}	F_{00}
$w_i + w'_i$	$F_{1.}$	$F_{0.}$

Semisupervised classification

The estimated score $\hat{S}$ of word w_i for class-label (country in our case) c_j is then estimated by focusing on **co-occurrences of words**

In particular $\hat{S}_{ij}$ (i.e., the "association score of word *i* for label *j*)" is calculated as the association between w_i and c_j subtracted by the association between w_i and $\overline{C}_j$:

$$\hat{S}_{ij} = log\frac{F_{11}}{F_{1.}} - log\frac{F_{01}}{F_{0.}}$$

Semisupervised classification

This table shows fictional scores given to the words most strongly associated with Ukraine and Iraq

	Ukraine	Score	Iraq	Score
1	Ukraine	11.84	Iraq	11.58
2	Ukrainian	10.36	Baghdad	10.56
3	Kiev	10.34	Iraqi	10.39
4	Ukrainians	7.94	Iraqis	8.15
5	Poroshenko	7.64	Anbar	8.14
6	Mariupol	7.15	Ramadi	7.55
7	Yatseniuk	6.94	Fallujah	7.51
8	Donetsk	6.92	Falluja	7.50
9	Slovyansk	6.84	Kirkuk	7.42
10	Lugansk	6.72	Tikrit	7.36

Many new words are identified based on **co-occurrences** both for Ukraine and Iraq! These words will be added to the **dictionary** you created starting from the original list of seed-words

Semisupervised classification

The classification-stage

Newsmap then predicts the class-label (i.e., $\hat{C}$ - countries in our case) most strongly associated with documents simply by finding a class-label that yields the largest total scores $\hat{S}$ weighted by the normalized frequency of word f_i in documents:

$$\hat{C} = \arg\max_j \sum_i \sum_j \hat{s}_{ij} f_i .$$

i.e., a given document will be assigned to Ukraine rather than Iraq if it presents words most frequently used in documents that use one of the seed-words related to Ukraine {Ukraine, Ukrainian*, Kiev} , even if it does not include **any of these seed-words**!

Semisupervised classification

This approach is advantageous because:

(1) training of new models **does not require** any manual coding but only a **seed word dictionary** used to train semisupervised document classifiers

(2) searching a corpus for dictionary words is not computationally intensive: creating a seed word dictionary for semisupervised models is easier because the number of words required for a seed word dictionary is a fraction of a keyword dictionary

(3) dictionaries can be ported to **different projects** without or little modification

(4) Finally, and contrary to a fully unsupervised model, you know ex-ante the content of the topics you are looking for!

Semisupervised classification

However:

- ✓ use of semisupervised models requires knowledge of both methodology and substance of what you are analyzing!

In particular, your seed words should be of high quality! And should be (reasonably) present in your corpus

Semisupervised classification

For some suggestions on how to improve a seed-words dictionary list, see Watanabe and Zhou (2020)

For example seed word dictionaries should be constructed **based primarily on theory**, but they could also include **frequent words** in the corpus to produce good classification results

Shusei et al. (2020), however, argue to be cautious about this latter option, because that would imply analyzing the same data as the one used to derive the seed words

Semisupervised classification

Moreover:

- ✓ Newsmap also requires users to define **all the relevant topics** in a seed dictionary because it estimates features' association with a topic by comparing between their frequencies in documents with the label and all other labels, **ignoring documents without labels**

So, for example, if in your corpus you have news discussing about Ukraine, Iraq, but also Italy, then if you have not identified seed-words about Italy as well, Newsmap will be ignore such documents in its training-stage!

Semisupervised classification

Moreover:

- ✓ Newsmap assigns one single topic to each single text, rather than assuming that each text is a *mixture of topics*. Under some given circumstances, as we already discussed with respect to topic models, this could be a limit

Semisupervised classification

An alternative is employing a semi-supervised topic model (Eshimay et al. 2020; Curini and Vignoli 2020)

The main idea: we assume that in our corpus there are two types of topics

- ✓ **topics with keywords** (or seed words using the jargon of Newsmap) defined ex-ante by the researcher, which are of primary interest to researchers and are referred to as *keyword topics*

For example, going back to our previous discussion, we can have 2 *keyword topics*:

a topic related to Ukraine defined by the keywords {Ukraine, Ukrainian*, Kiev} and a topic about Iraq defined by the keywords {Iraq, Iraqi*, Baghdad}

Semisupervised classification

And..

✓ **topics without keywords**, called *no-keyword topics*

One key-difference with Newsmap is that you can have therefore one or more of off-kewyords topics (beyond the ones you identify via your seed-words selection). And you can still recover them from the analysis!

Note that the algorithm we will employ is generally robust to the number of off-keywords topics (as long as they are >1)

Semisupervised classification

How it works? Going back to our example, let's suppose that we select k=3, with 2 keyword topics (Ukraine and Iraq) and 1 no-keyword topic

A semi-supervised topic model is a traditional topic model with the following difference in the **third step of the procedure** (i.e., after having extracted θ_i and β_k)

First, as usual, you draw a latent topic from θ_i

If the sampled topic is one of the no-keyword topics, then we draw word w_{im} (the *m*th word in document *i*) from the corresponding word distribution of the topic. And we follow our usual approach

Semisupervised classification

However, if the sampled topic is a keyword topics (i.e., either Ukraine or Iraq), we first draw a Bernoulli random variable with success probability *p* for word *m* in document *i*

If this variable is equal to 1, then word w_{im} is drawn from the set of keywords for the topic that we defined ex-ante

In contrast, if the Bernoulli random variable is equal to 0, then we sample the word from the standard topic-word distribution of the topic

Semisupervised classification

In other words, the semi-supervised topic is based on a *mixture of two distributions*, one with positive probabilities only for keywords and the other with positive probabilities for all words

It is straightforward to show that this mixture structure makes the prior means for the frequency of user-selected keywords (i.e., seed words) given a topic greater than those of non-keywords in the same topic

In addition, the prior variance is also larger for the frequency of keywords given a topic than for non-keywords

This encourages the algorithm to place a greater importance on keywords a priori while allowing the model to learn from the data about the precise degree to which keywords matter for a given topic

Semisupervised classification

As a result, each topic with keywords already has a label and so there is no need to interpret the resulting topics after model fitting (one topic is going to be Ukraine and the other is going to be Iraq)

In contrast, the no-keywords topics require as usual a post-hoc labeling (could it be related to Italy?)

Semisupervised classification

The selection of **high quality keywords** is once again critical for the successful application of such approach!

1. The keywords selection should be theoretically sound! That also implies that (within each topic) keywords should refer to the same topic!
2. The keywords should be present in a non-negligible way in the corpus (this is something you can **check ex-ante**)
3. Moreover, the unique keywords selected for each topic should appear frequently in the keyword topic, i.e., they should discriminate their topics from others (this is something you can **check only ex-post**)

Note that the same keywords may be used for different keyword topics and keywords are a part of the total corpus unique words

Semisupervised classification

Interestingly, you can incorporate within a semi-supervised topic model also covariates for the document-topic distribution as social scientists often have meta information about documents

This is similar to what we discussed for STM. Do you remember?

We can have, in other words, a semisupervised *structural* topic algorithm!

R pakcages to install

```
install.packages("wordcloud", repos='http://cran.us.r-project.org')
install.packages("dplyr", repos='http://cran.us.r-project.org')
install.packages("gridExtra", repos='http://cran.us.r-project.org')
install.packages("syuzhet", repos='http://cran.us.r-project.org')
install.packages ("reshape2", repos='http://cran.us.r-project.org')
install.packages ("tm", repos='http://cran.us.r-project.org')
install.packages ("plyr", repos='http://cran.us.r-project.org')
install.packages("newsmap", repos='http://cran.us.r-project.org')
install.packages("magrittr", repos='http://cran.us.r-project.org')
devtools::install_github("keyATM/keyATM", ref = "v0.4.0")
```