Session 2
Topics models and machine-learning algorithms for classification

Michele Scotto di Vettimo
King's College London

https://mscottodivettimo.github.io/
michele.scotto_di_vettimo@kcl.ac.uk

LISS2117 · *Quantitative methods for text classification and topic detection*

Programme

- Unsupervised classification: topic models
- Semisupervised classification (1): Keyword assisted topic models
- Supervised classification (2): Machine learning algorithms
- Validation

Recap of previous session

In previous episodes...

- Text-as-data methods builds on numerical representation of our raw texts

In previous episodes...

- Text-as-data methods builds on numerical representation of our raw texts
- Bag-of-words representations are the most common way of representing texts into numerical form

In previous episodes...

- Text-as-data methods builds on numerical representation of our raw texts
- Bag-of-words representations are the most common way of representing texts into numerical form
  - They do so by splitting texts into tokens, and then arranging them in a document-feature matrix (dfm)

In previous episodes...

- Text-as-data methods builds on numerical representation of our raw texts
- Bag-of-words representations are the most common way of representing texts into numerical form
  - They do so by splitting texts into tokens, and then arranging them in a document-feature matrix (dfm)
  - The construction of the dfm needs to be tailored to the research task

In previous episodes...

- Text-as-data methods builds on numerical representation of our raw texts
- Bag-of-words representations are the most common way of representing texts into numerical form
  - They do so by splitting texts into tokens, and then arranging them in a document-feature matrix (dfm)
  - The construction of the dfm needs to be tailored to the research task
- Classification methods rely on the mapping of tokens into categories/labels

In previous episodes...

- Text-as-data methods builds on numerical representation of our raw texts
- Bag-of-words representations are the most common way of representing texts into numerical form
  - They do so by splitting texts into tokens, and then arranging them in a document-feature matrix (dfm)
  - The construction of the dfm needs to be tailored to the research task
- Classification methods rely on the mapping of tokens into categories/labels
  - In dictionary methods, the analyst provides this mapping beforehand, whereas in other methods it is "learned" from the data

In previous episodes...

- Text-as-data methods builds on numerical representation of our raw texts
- Bag-of-words representations are the most common way of representing texts into numerical form
  - They do so by splitting texts into tokens, and then arranging them in a document-feature matrix (dfm)
  - The construction of the dfm needs to be tailored to the research task
- Classification methods rely on the mapping of tokens into categories/labels
  - In dictionary methods, the analyst provides this mapping beforehand, whereas in other methods it is "learned" from the data
- Validation is an essential part of the analysis, and different methods require different types of validation

In previous episodes...

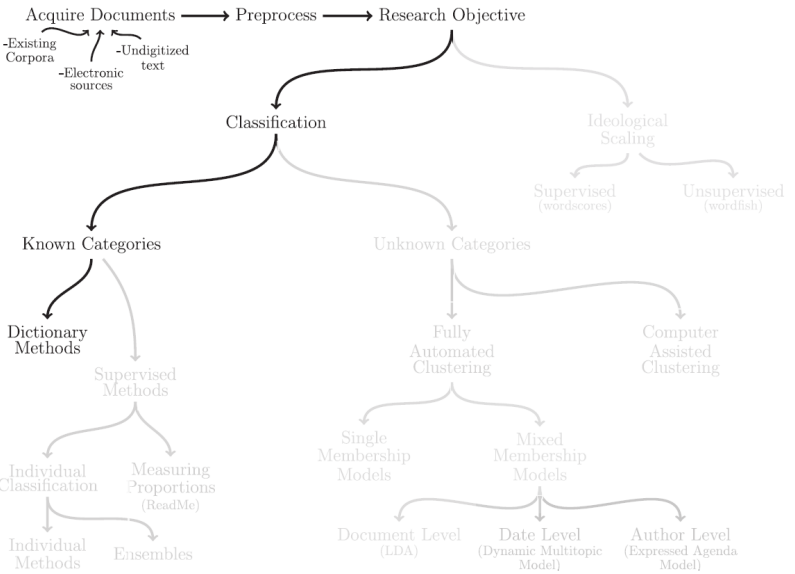


Fig. 1 An overview of text as data methods.

Unsupervised classification

Unsupervised, semisupervised, and supervised methods

We use these terms to categorise methods according to how much labelled data (human guidance, basically) is needed to learn relations between tokens and labels (and ultimately to assign a text to a class)

Unsupervised, semisupervised, and supervised methods

We use these terms to categorise methods according to how much labelled data (human guidance, basically) is needed to learn relations between tokens and labels (and ultimately to assign a text to a class)

In the previous session we covered dictionary methods. In dictionaries the association between tokens and labels is specified in advance, so the model does not do any learning. Hence, dictionary methods do not really fit this categorisation.

Unsupervised, semisupervised, and supervised methods

We use these terms to categorise methods according to how much labelled data (human guidance, basically) is needed to learn relations between tokens and labels (and ultimately to assign a text to a class)

In the previous session we covered dictionary methods. In dictionaries the association between tokens and labels is specified in advance, so the model does not do any learning. Hence, dictionary methods do not really fit this categorisation.

However, in other cases text-as-data models learn associations between tokens in our texts and labels from the corpus, and they might do so with the help of some labelled data.

Unsupervised, semisupervised, and supervised methods

We use these terms to categorise methods according to how much labelled data (human guidance, basically) is needed to learn relations between tokens and labels (and ultimately to assign a text to a class)

In the previous session we covered dictionary methods. In dictionaries the association between tokens and labels is specified in advance, so the model does not do any learning. Hence, dictionary methods do not really fit this categorisation.

However, in other cases text-as-data models learn associations between tokens in our texts and labels from the corpus, and they might do so with the help of some labelled data.

> **Unsupervised methods**: use modeling assumptions and properties of the texts to estimate categories and assign documents to them (no labelled data needed)

Unsupervised, semisupervised, and supervised methods

We use these terms to categorise methods according to how much labelled data (human guidance, basically) is needed to learn relations between tokens and labels (and ultimately to assign a text to a class)

In the previous session we covered dictionary methods. In dictionaries the association between tokens and labels is specified in advance, so the model does not do any learning. Hence, dictionary methods do not really fit this categorisation.

However, in other cases text-as-data models learn associations between tokens in our texts and labels from the corpus, and they might do so with the help of some labelled data.

Unsupervised methods: use modeling assumptions and properties of the texts to estimate categories and assign documents to them (no labelled data needed)

Semisupervised methods: use both a small amount of labelled data and unlabelled texts to learn relations between tokens and labels and classify texts (limited amount of labelled data used)

Unsupervised, semisupervised, and supervised methods

We use these terms to categorise methods according to how much labelled data (human guidance, basically) is needed to learn relations between tokens and labels (and ultimately to assign a text to a class)

In the previous session we covered dictionary methods. In dictionaries the association between tokens and labels is specified in advance, so the model does not do any learning. Hence, dictionary methods do not really fit this categorisation.

However, in other cases text-as-data models learn associations between tokens in our texts and labels from the corpus, and they might do so with the help of some labelled data.

Unsupervised methods: use modeling assumptions and properties of the texts to estimate categories and assign documents to them (no labelled data needed)

Semisupervised methods: use both a small amount of labelled data and unlabelled texts to learn relations between tokens and labels and classify texts (limited amount of labelled data used)

Supervised methods: trained on labelled texts to learn relations between tokens and labels and classify texts (labelled training data required)

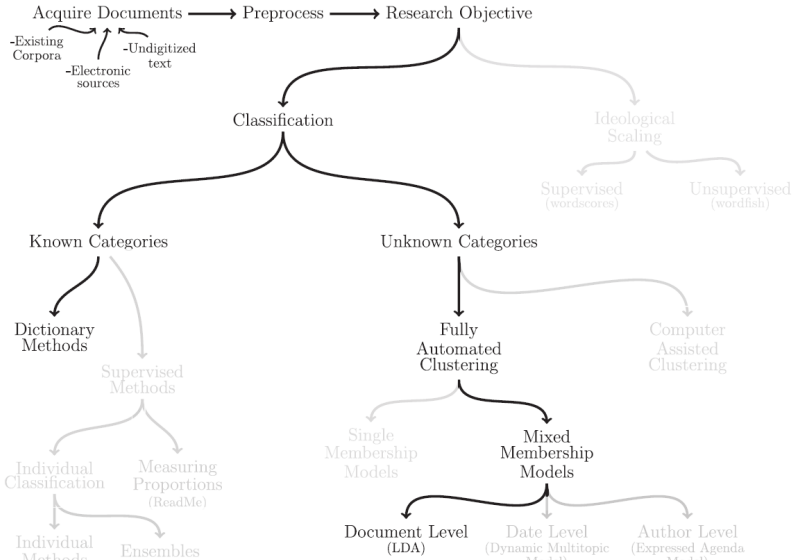


Fig. 1 An overview of text as data methods.

Topic models

Topic models are a class of models that assign each text with proportional membership to all categories ("topics").

Each text is treated as a mixture of categories (that is why we categorise these models as "mixed membership models")

Topic models

Topic models are a class of models that assign each text with proportional membership to all categories ("topics").

Each text is treated as a mixture of categories (that is why we categorise these models as "mixed membership models")

One document could be assigned to just a single category, but it does not need to be so. This makes results more "flexible" and humanly interpretable

Topic models

Topic models are a class of models that assign each text with proportional membership to all categories ("topics").

Each text is treated as a mixture of categories (that is why we categorise these models as "mixed membership models")

One document could be assigned to just a single category, but it does not need to be so. This makes results more "flexible" and humanly interpretable

When we talk simply of "topic models", we generally mean unsupervised models that assign texts to a set of unlabelled, not predetermined, categories

Topic models

Topic model with 10 topics implemented on our BBC News 2023 corpus

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
text1	0.075	0.000	0.020	0.083	0.000	0.169	0.644	0.000	0.008	0.000
text2	0.004	0.000	0.008	0.571	0.000	0.000	0.019	0.000	0.397	0.000
text3	0.003	0.000	0.006	0.014	0.841	0.000	0.000	0.067	0.061	0.008
text4	0.000	0.000	0.000	0.000	0.064	0.000	0.927	0.007	0.000	0.000
text5	0.000	0.313	0.000	0.004	0.090	0.000	0.000	0.313	0.000	0.279
text6	0.000	0.000	0.037	0.000	0.002	0.931	0.028	0.000	0.000	0.000
text7	0.001	0.477	0.001	0.001	0.006	0.001	0.062	0.324	0.001	0.129
text8	0.000	0.054	0.017	0.000	0.014	0.000	0.000	0.000	0.027	0.886
text9	0.001	0.031	0.103	0.001	0.001	0.021	0.713	0.036	0.093	0.001
text10	0.045	0.000	0.080	0.000	0.000	0.040	0.000	0.758	0.005	0.070
text11	0.000	0.841	0.000	0.000	0.012	0.026	0.000	0.000	0.093	0.026
text12	0.000	0.000	0.000	0.165	0.004	0.004	0.657	0.000	0.165	0.004
text13	0.245	0.016	0.001	0.059	0.011	0.665	0.001	0.001	0.001	0.001
text14	0.072	0.001	0.308	0.001	0.158	0.108	0.329	0.022	0.001	0.001
text15	0.000	0.216	0.145	0.017	0.000	0.000	0.000	0.399	0.156	0.066
text16	0.000	0.000	0.000	0.000	0.832	0.086	0.004	0.000	0.000	0.077
text17	0.026	0.026	0.000	0.004	0.012	0.000	0.000	0.000	0.000	0.930
text18	0.028	0.016	0.063	0.000	0.491	0.075	0.000	0.071	0.000	0.255
text19	0.615	0.001	0.042	0.012	0.012	0.001	0.036	0.281	0.001	0.001
text20	0.000	0.000	0.000	0.055	0.774	0.168	0.000	0.000	0.000	0.000

Distributions of the 10 topics for 20 documents in our corpus

Topic models

Topic model with 10 topics implemented on our BBC News 2023 corpus

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
1	us	none	king	mr	pay	uk	us	police	mr	content
2	bank	first	visit	minister	health	government	russian	mr	lineker	external
3	mr	world	prince	ms	staff	energy	ukraine	court	former	twitter
4	ireland	game	uk	party	school	new	russia	ms	sir	sites
5	northern	england	president	snp	government	tax	mr	found	video	browser
6	banks	league	first	first	nhs	years	president	family	day	responsible
7	uk	club	british	government	members	work	city	officers	public	post
8	deal	football	france	yousaf	action	now	war	man	case	view
9	financial	united	government	leader	strike	help	ukrainian	heard	secretary	just
10	company	manchester	royal	scottish	children	get	military	car	media	video
11	new	right	french	forbes	care	prices	video	attack	need	snow
12	data	win	charles	scotland	offer	food	defence	home	made	policy
13	eu	players	state	prime	union	water	bakhmut	video	government	javascript
14	money	team	years	sturgeon	services	cost	trump	died	inquiry	enable
15	credit	time	queen	vote	schools	plans	forces	time	evidence	like
16	twitter	cup	city	secretary	england	added	state	years	rules	weather
17	ai	side	day	labour	teachers	change	nuclear	house	messages	want
18	since	won	macron	leadership	workers	time	since	murder	policy	please
19	musk	second	video	election	strikes	costs	putin	case	social	anything
20	government	top	part	johnson	education	budget	security	just	gary	see

Top 20 words, ordered by frequency, for the estimated 10 topics

Latent Dirichlet Allocation (LDA)

The most common topic model is Latent Dirichlet Allocation (LDA) [Blei et al., 2003]

Latent Dirichlet Allocation (LDA)

The most common topic model is Latent Dirichlet Allocation (LDA) [Blei et al., 2003]

LDA goal is to retrieve the latent distribution of topic proportions across documents, starting from the texts and a formalisation of how they are generated.

Latent Dirichlet Allocation (LDA)

The most common topic model is Latent Dirichlet Allocation (LDA) [Blei et al., 2003]

LDA goal is to retrieve the latent distribution of topic proportions across documents, starting from the texts and a formalisation of how they are generated.

It works as follows:

We observe only the words in our documents

Doc1: white red blue violet
Doc2: president america white house

Latent Dirichlet Allocation (LDA)

The most common topic model is Latent Dirichlet Allocation (LDA) [Blei et al., 2003]

LDA goal is to retrieve the latent distribution of topic proportions across documents, starting from the texts and a formalisation of how they are generated.

It works as follows:

We observe only the words in our documents

Doc1: white red blue violet

Doc2: president america white house

And set the number of topics we want the model to retrieve

$k = 2$

Latent Dirichlet Allocation (LDA)

The most common topic model is Latent Dirichlet Allocation (LDA) [Blei et al., 2003]

LDA goal is to retrieve the latent distribution of topic proportions across documents, starting from the texts and a formalisation of how they are generated.

It works as follows:

We observe only the words in our documents

Doc1: white red blue violet

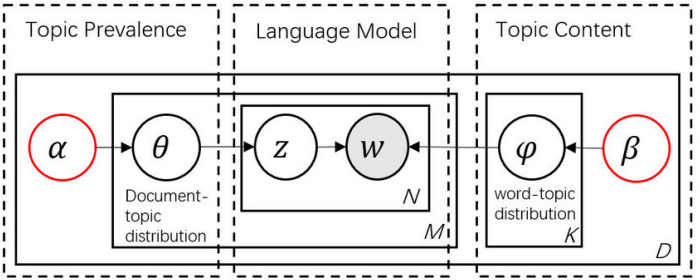
Doc2: president america white house

And set the number of topics we want the model to retrieve

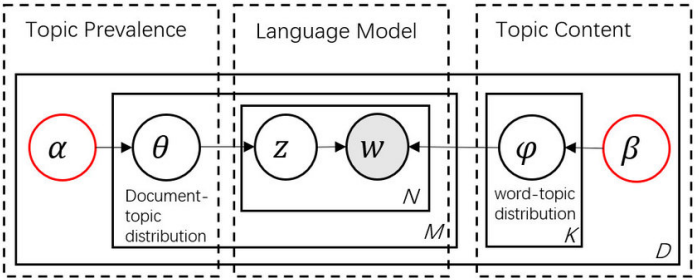
$k = 2$

Then, a statistical model formalises our understanding of how texts are generated.

Latent Dirichlet Allocation (LDA)

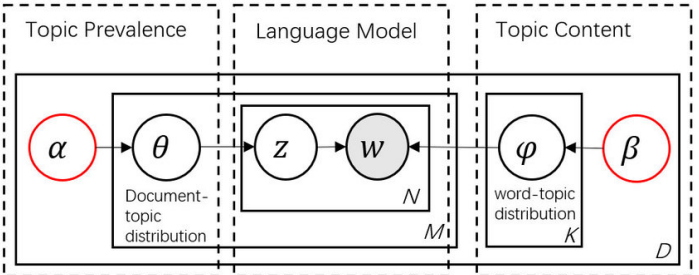


Latent Dirichlet Allocation (LDA)



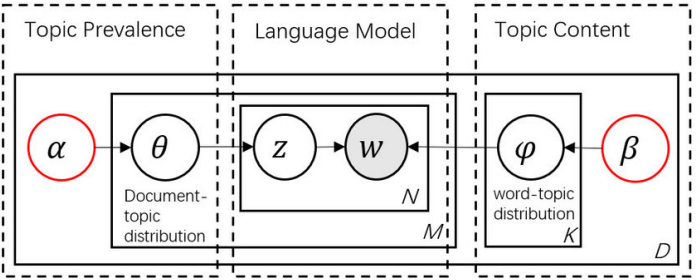
- α controls the distribution of a topic in a document m, represented by θ.

Latent Dirichlet Allocation (LDA)



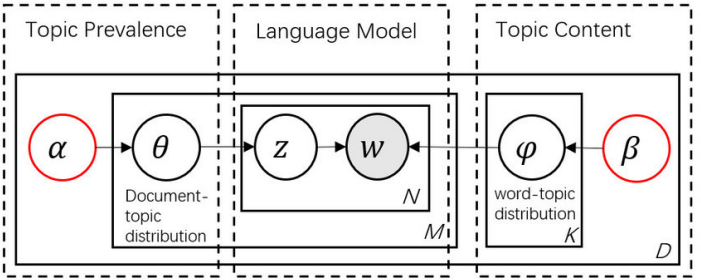
- α controls the distribution of a topic in a document m, represented by θ.
- β influences how words are distributed across topics k, represented by ϕ.

Latent Dirichlet Allocation (LDA)



- α controls the distribution of a topic in a document m, represented by θ.
- β influences how words are distributed across topics k, represented by ϕ.
- z represents the topic assignment of word w in document m, given our topics.

Latent Dirichlet Allocation (LDA)



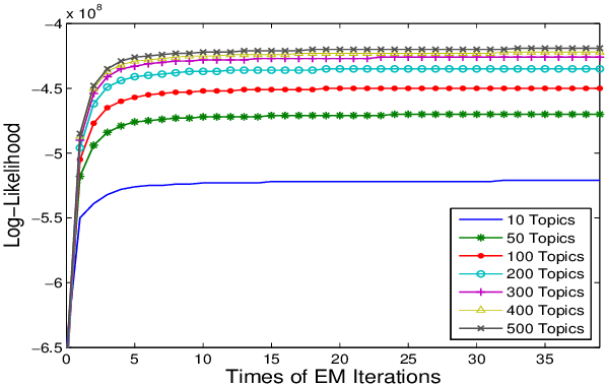
- α controls the distribution of a topic in a document m, represented by θ.
- β influences how words are distributed across topics k, represented by ϕ.
- z represents the topic assignment of word w in document m, given our topics.
- However, we observe only w and K, but we are interested in θ and ϕ

Latent Dirichlet Allocation (LDA)

A Bayesian model tries to reverse engineer the process by guessing different ways of grouping words into the k topics.

At each step, the model measures how likely is it that the topic assignments could've generated the observed documents, and adapts them so as to increase this likelihood.

Iterations continue until we reach a point where our measure of likelihood/statistical fit does not improve any more. Model has converged, and α, β, θ, and ϕ are computed, given the topic assignments.



Latent Dirichlet Allocation (LDA)

Doc1: white red blue violet

Doc2: president america white house

$k = 2$

Latent Dirichlet Allocation (LDA)

Doc1: white red blue violet

Doc2: president america white house

$k = 2$

1. LDA produces a random first assignment of words to topics:
   Doc1: white^{topic1} red^{topic1} blue^{topic2} violet^{topic1}
   Doc2: president^{topic2} america^{topic1} white^{topic2} house^{topic2}

Latent Dirichlet Allocation (LDA)

Doc1: white red blue violet

Doc2: president america white house

$k = 2$

1. LDA produces a random first assignment of words to topics:
   Doc1: white^{topic1} red^{topic1} blue^{topic2} violet^{topic1}
   Doc2: president^{topic2} america^{topic1} white^{topic2} house^{topic2}
2. Assess fit using log-likelihood measure
   - How likely is it that these topic assignments could've generated the observed documents?
   - Maximize the joint probability of all the words in the documents, topic assignments, and topic-word and document-topic distributions.

Latent Dirichlet Allocation (LDA)

3. LDA samples a new word-topic assignment for our documents:
   Doc1: white^{topic1} red^{topic1} blue^{topic1} violet^{topic1}
   Doc2: president^{topic2} america^{topic1} white^{topic2} house^{topic2}

Latent Dirichlet Allocation (LDA)

3. LDA samples a new word-topic assignment for our documents:
   Doc1: white^{topic1} red^{topic1} blue^{topic1} violet^{topic1}
   Doc2: president^{topic2} america^{topic1} white^{topic2} house^{topic2}
4. Assess fit once again. Has our measure of fit (log-likelihood) improved?

Latent Dirichlet Allocation (LDA)

3. LDA samples a new word-topic assignment for our documents:
   Doc1: white^{topic1} red^{topic1} blue^{topic1} violet^{topic1}
   Doc2: president^{topic2} america^{topic1} white^{topic2} house^{topic2}
4. Assess fit once again. Has our measure of fit (log-likelihood) improved?
   - Yes: go ahead with the iterations.
   - No or not much or we've reached the limit of our iterations: model stops.

Latent Dirichlet Allocation (LDA)

3. LDA samples a new word-topic assignment for our documents:

   Doc1: white^{topic1} red^{topic1} blue^{topic1} violet^{topic1}

   Doc2: president^{topic2} america^{topic1} white^{topic2} house^{topic2}

4. Assess fit once again. Has our measure of fit (log-likelihood) improved?
   - Yes: go ahead with the iterations.
   - No or not much or we've reached the limit of our iterations: model stops.

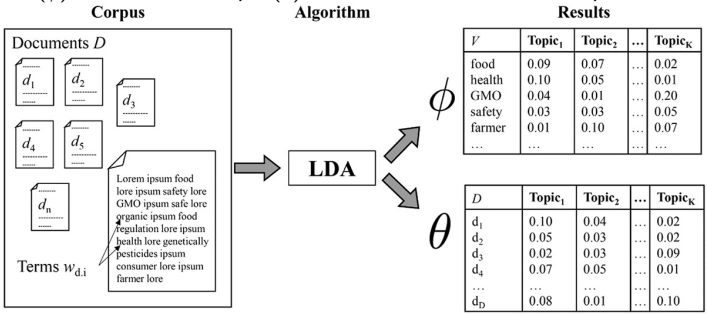
     These are the final topic-word assignments. LDA now computes the parameters of the topic-word (ϕ) and document-topic (θ) distributions from the word-topic counts.

Latent Dirichlet Allocation (LDA)

3. LDA samples a new word-topic assignment for our documents:
   Doc1: white^{topic1} red^{topic1} blue^{topic1} violet^{topic1}
   Doc2: president^{topic2} america^{topic1} white^{topic2} house^{topic2}

4. Assess fit once again. Has our measure of fit (log-likelihood) improved?
   - Yes: go ahead with the iterations.
   - No or not much or we've reached the limit of our iterations: model stops.

   These are the final topic-word assignments. LDA now computes the parameters of the topic-word (ϕ) and document-topic (θ) distributions from the word-topic counts.



V	Topic$_1$	Topic$_2$	…	Topic$_K$
food	0.09	0.07	…	0.02
health	0.10	0.05	…	0.01
GMO	0.04	0.01	…	0.20
safety	0.03	0.03	…	0.05
farmer	0.01	0.10	…	0.07
…	…	…	…	…

D	Topic$_1$	Topic$_2$	…	Topic$_K$
d_1	0.10	0.04	…	0.02
d_2	0.05	0.03	…	0.02
d_3	0.02	0.03	…	0.09
d_4	0.07	0.05	…	0.01
…	…	…	…	…
d_D	0.08	0.01	…	0.10

[Maier et al., 2018]

Topic models

Topic model with 10 topics implemented on our BBC News 2023 corpus

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
1	us	none	king	mr	pay	uk	us	police	mr	content
2	bank	first	visit	minister	health	government	russian	mr	lineker	external
3	mr	world	prince	ms	staff	energy	ukraine	court	former	twitter
4	ireland	game	uk	party	school	new	russia	ms	sir	sites
5	northern	england	president	snp	government	tax	mr	found	video	browser
6	banks	league	first	first	nhs	years	president	family	day	responsible
7	uk	club	british	government	members	work	city	officers	public	post
8	deal	football	france	yousaf	action	now	war	man	case	view
9	financial	united	government	leader	strike	help	ukrainian	heard	secretary	just
10	company	manchester	royal	scottish	children	get	military	car	media	video
11	new	right	french	forbes	care	prices	video	attack	need	snow
12	data	win	charles	scotland	offer	food	defence	home	made	policy
13	eu	players	state	prime	union	water	bakhmut	video	government	javascript
14	money	team	years	sturgeon	services	cost	trump	died	inquiry	enable
15	credit	time	queen	vote	schools	plans	forces	time	evidence	like
16	twitter	cup	city	secretary	england	added	state	years	rules	weather
17	ai	side	day	labour	teachers	change	nuclear	house	messages	want
18	since	won	macron	leadership	workers	time	since	murder	policy	please
19	musk	second	video	election	strikes	costs	putin	case	social	anything
20	government	top	part	johnson	education	budget	security	just	gary	see

Top 20 words, ordered by frequency, for the estimated 10 topics

Evaluating LDA models

LDA are unsupervised models and, therefore, require thorough ex post interpretation, evaluation and validation

Evaluating LDA models

LDA are unsupervised models and, therefore, require thorough ex post interpretation, evaluation and validation

- Human judgment
  - Top N words per topic (ϕ, frequency/exclusivity (FREX) score)
  - Document-topic distributions of "obvious" texts

Evaluating LDA models

LDA are unsupervised models and, therefore, require thorough ex post interpretation, evaluation and validation

- Human judgment
  - Top N words per topic (ϕ, frequency/exclusivity (FREX) score)
  - Document-topic distributions of "obvious" texts
- Evaluation metrics
  - Log-Likelihood to inspect model convergence
  - Model perplexity to measure how good the model is at predicting unseen (held out) data
  - Semantic coherence to assess how semantically close are the high scoring words in a topic

FREX score

The FREX measure is used to rank word, within each topic, according to their frequency and exclusivity to a topic.

It balances two goals: finding words that are used frequently within a topic, and words that are distinctive to that topic

FREX score

The FREX measure is used to rank word, within each topic, according to their frequency and exclusivity to a topic.

It balances two goals: finding words that are used frequently within a topic, and words that are distinctive to that topic

Empirically, it is the harmonic mean of two rankings:

- *frequency* rank (probability of the word under the topic)
- *exclusivity* rank (probability under a topic vs. other topics)

$$FREX = \frac{1}{\left(\frac{w}{frequency} + \frac{(1-w)}{exclusivity}\right)}$$

where w is just a weight (typically $= 0.7$) to balance the two rankings

FREX score

The FREX measure is used to rank word, within each topic, according to their frequency and exclusivity to a topic.

It balances two goals: finding words that are used frequently within a topic, and words that are distinctive to that topic

Empirically, it is the harmonic mean of two rankings:

- *frequency* rank (probability of the word under the topic)
- *exclusivity* rank (probability under a topic vs. other topics)

$$FREX = \frac{1}{\left(\frac{w}{frequency} + \frac{(1-w)}{exclusivity}\right)}$$

where w is just a weight (typically $= 0.7$) to balance the two rankings

High FREX values indicate words that are both frequent and distinctive to one topic

Model perplexity

Model perplexity is a measure used to evaluate how well a topic model predicts the topic-document distribution in a new sample.

Roughly speaking, perplexity is measured by running an LDA model to estimate the various parameters describing the data-generating process, and then assessing how likely are the words in the document given the estimated parameters.

Model perplexity

Model perplexity is a measure used to evaluate how well a topic model predicts the topic-document distribution in a new sample.

Roughly speaking, perplexity is measured by running an LDA model to estimate the various parameters describing the data-generating process, and then assessing how likely are the words in the document given the estimated parameters.

Formally, it is defined as the exponential of the negative average log-likelihood per word in the documents.

Model perplexity

Model perplexity is a measure used to evaluate how well a topic model predicts the topic-document distribution in a new sample.

Roughly speaking, perplexity is measured by running an LDA model to estimate the various parameters describing the data-generating process, and then assessing how likely are the words in the document given the estimated parameters.

Formally, it is defined as the exponential of the negative average log-likelihood per word in the documents.

Low perplexity values indicate better fit. However,

- it is sensitive to the size of the vocabulary, hence you should be careful when using it for model comparison
- it does not say much about topic quality or human interpretability

Semantic coherence

Semantic coherence measures the semantic similarity between high scoring words in each topic. Conceptually, it assesses whether the words making up a topic are really about similar things.

Therefore, it can distinguish interpretable topics from those that are just a statistical artefact. In this sense, tries to capture topic-quality beyond a statistical fit like perplexity

Semantic coherence

Semantic coherence measures the semantic similarity between high scoring words in each topic. Conceptually, it assesses whether the words making up a topic are really about similar things.

Therefore, it can distinguish interpretable topics from those that are just a statistical artefact. In this sense, tries to capture topic-quality beyond a statistical fit like perplexity

In practice, it looks at how often the top words in a topic co-occur in the documents to approximate their semantic similarity, and then get a measure of overall topic coherence

Semantic coherence

Semantic coherence measures the semantic similarity between high scoring words in each topic. Conceptually, it assesses whether the words making up a topic are really about similar things.

Therefore, it can distinguish interpretable topics from those that are just a statistical artefact. In this sense, tries to capture topic-quality beyond a statistical fit like perplexity

In practice, it looks at how often the top words in a topic co-occur in the documents to approximate their semantic similarity, and then get a measure of overall topic coherence

High coherence values indicate good topics. Also:

- it is way less sensitive than perplexity to changes in the vocabulary, so more useful for model comparison
- average topic coherence is generally used as metric to evaluate different LDA models (e.g., by plotting coherence vs k)

We will be mostly relying on the `textmineR` package:

- `FitLdaModel()`
- other functions for post-estimation

We will also keep using numerous `quanteda` functions introduced in the previous session.

When lost, cry for `help()`!

Structural Topic Models (STM)

In LDA setting, all documents are assumed to come from the same data-generating process: all documents' mixture of topics is drawn from the same distribution

However, there cases where topic proportions or topic content are influenced by some document-level factors:

- e.g., news articles from 2020 talk more about "pandemics" than articles from 2010 (time influences topic prevalence)
- e.g., newspapers with different audiences use different words to talk about the same topics ("cost of living" vs "inflation") (newspaper type influences topic content)

Structural Topic Models (STM)

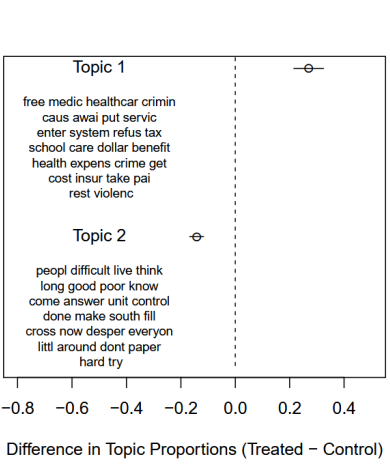
In LDA setting, all documents are assumed to come from the same data-generating process: all documents' mixture of topics is drawn from the same distribution

However, there cases where topic proportions or topic content are influenced by some document-level factors:

- e.g., news articles from 2020 talk more about "pandemics" than articles from 2010 (time influences topic prevalence)
- e.g., newspapers with different audiences use different words to talk about the same topics ("cost of living" vs "inflation") (newspaper type influences topic content)

Structural Topic Models (STM) [Roberts et al., 2016] can incorporate the information from document-level metadata to better model topic prevalence and content, which are allowed to vary based on document's characteristics

Structural Topic Models (STM)



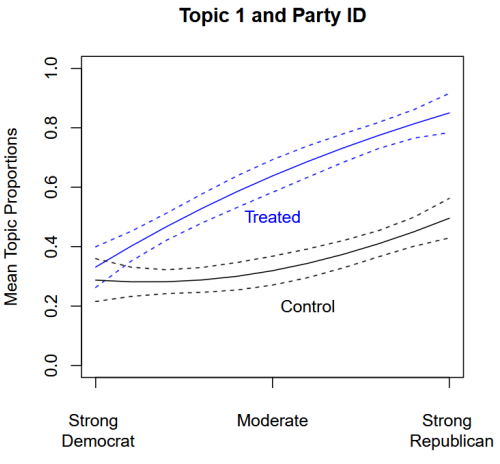
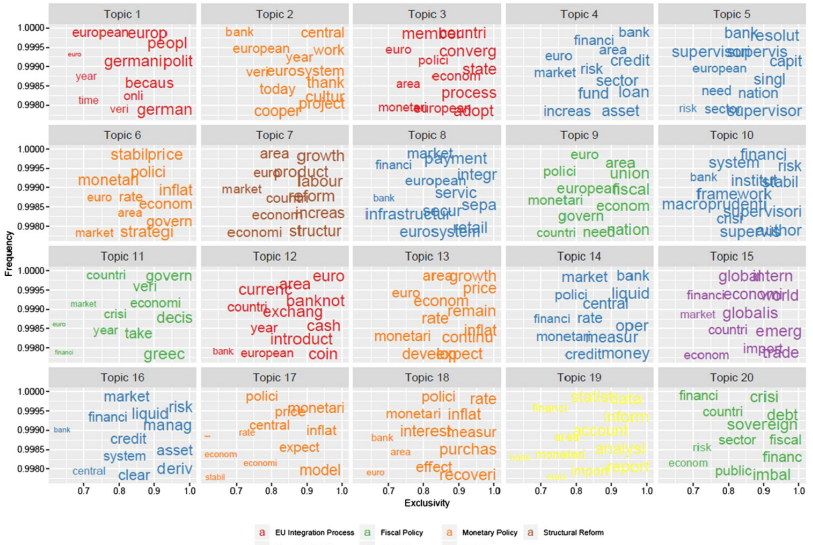


Figure 2: Party ID, Treatment, and the Predicted Proportion in Fear Topic (1 of 3)

[Roberts et al., 2014]

Structural Topic Models (STM)



[Moschella et al., 2020]

Structural Topic Models vs LDA

Whether you should prefer STM over LDA depends on your research goals and on the nature of your data

Structural Topic Models vs LDA

Whether you should prefer STM over LDA depends on your research goals and on the nature of your data

STM can be preferable if

- You have interesting document-level metadata
- You want to do hypothesis testing or regression analysis on topics
- You want to model changes over time or across groups

Structural Topic Models vs LDA

Whether you should prefer STM over LDA depends on your research goals and on the nature of your data

STM can be preferable if

- You have interesting document-level metadata
- You want to do hypothesis testing or regression analysis on topics
- You want to model changes over time or across groups

LDA is the way to go if

- You do not have metadata or you are just not interested in them
- You do not want to make assumptions on topics based on document metadata
- Your corpus is very large and you value speed over topic variation by metadata

Semisupervised classification

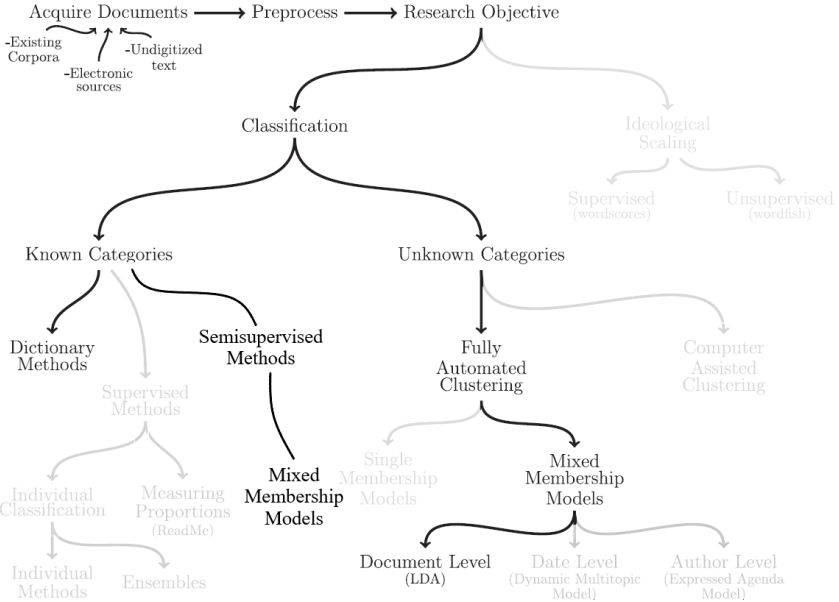


Fig. 1 An overview of text as data methods.

Keyword-Assisted Topic Models (KeyATM)

There are topic models that try to combine the features of dictionary methods (having keywords to guide topic-word distributions) and unsupervised methods (learning from word co-occurrence the topic-document distributions)

Because of the use of keywords, such models are called semi-supervised

Keyword-Assisted Topic Models (KeyATM)

There are topic models that try to combine the features of dictionary methods (having keywords to guide topic-word distributions) and unsupervised methods (learning from word co-occurrence the topic-document distributions)

Because of the use of keywords, such models are called semi-supervised

One example are Keyword-Assisted Topic Models (KeyATM) [Eshima et al., 2024]

However, there are other implementations that do very similar things (e.g., Seeded LDA [Watanabe and Zhou, 2020])

Keyword-Assisted Topic Models (KeyATM)

TABLE 3 Keywords for Each Topic

Type	Topic Label	Keywords
Pork barrel	Public works	employment, public, works
	Road construction	road, budget
Programmatic	Regional devolution	rural area, devolve, merger
	Tax	consumption, tax, tax increase
	Economic recovery	economic climate, measure, fiscal policy, deficit
	Global economy	trade, investment, industry
	Alternation of government	government, alternation
	Constitution	constitution
	Party	party, political party
	Postal privatization	postal, privatize
	Inclusive society	women, participate, civilian
	Social welfare	society, welfare
	Pension	pension
	Education	education
	Environment	environment, protection
	Security	defense, foreign policy, self defense

Notes: The left and middle columns show the types of policies and topic labels assigned by Catalinac (2016). The corresponding keywords in the right column are obtained from the UTokyo-Asahi Surveys (UTAS). This results in the removal of five policy areas (sightseeing, regional revitalization, policy vision, political position, and investing more on human capital) that do not appear in the UTAS.

[Eshima et al., 2024]

Keyword-Assisted Topic Models (KeyATM)

TABLE 4 Comparison of Top 10 Words for Six Selected Topics between the Covariate keyATM and STM

Road Construction		Tax		Economic Recovery	
keyATM	STM	keyATM	STM	keyATM	STM
development	tax	Japan	Japan	reform	reform
road	reduce tax	**tax**	citizen	**measure**	postal
city	yen	citizen	JCP	society∗	privatize
construction	housing	JCP	politic	Japan	Japan
tracks	realize	**consumption**	tax	**economic climate**	rural area
budget	daily life	politic	consumption	reassure	country
realize	move forward	**tax increase**	tax increase	economy	citizen
promote	city	oppose	oppose	institution	safe
move forward	education	business	business	safe	government
early	measure	protect	protect	support	pension

Inclusive Society		Education		Security	
keyATM	STM	keyATM	STM	keyATM	STM
politic	politic	politic	Japan	Japan	society
civilian	reform	Japan	person	**foreign policy**	Japan
society∗	new	person	country	peace	world
participate	realize	**children**	politic	world	economy
peace	citizen	**education**	necessary	economy	environment
welfare∗	government	country	problem	country	international
aim	daily life	make	children	citizen	education
human rights	rural area	force	force	**defense**	country
realize	corruption	have	have	safe	peace
consumption∗	change	problem	future	international	aim

Notes: The table shows the ten words with the highest estimated probabilities for each topic under each model. For keyATM, the pre-specified keywords for each topic appear in bold letters whereas the asterisks indicate the keywords specified for another topic.

[Eshima et al., 2024]

Keyword-Assisted Topic Models (KeyATM)

KeyATM enhances traditional topic modeling (like LDA) and STM by allowing users to provide seed words (keywords) for some topics.

Keyword-Assisted Topic Models (KeyATM)

KeyATM enhances traditional topic modeling (like LDA) and STM by allowing users to provide seed words (keywords) for some topics.

The model then:

- Softly biases specific topics to prefer those keywords - but does not fix them (keywords could end up in other topics)
- Allows other words to join these topics based on co-occurrence with the keywords
- Other topics can be left unguided and are discovered in a fully unsupervised way
- Optionally, KeyATM can model how covariates (e.g., author, time) influence topic prevalence - like in STM

Keyword-Assisted Topic Models (KeyATM)

KeyATM enhances traditional topic modeling (like LDA) and STM by allowing users to provide seed words (keywords) for some topics.

The model then:

- Softly biases specific topics to prefer those keywords - but does not fix them (keywords could end up in other topics)
- Allows other words to join these topics based on co-occurrence with the keywords
- Other topics can be left unguided and are discovered in a fully unsupervised way
- Optionally, KeyATM can model how covariates (e.g., author, time) influence topic prevalence - like in STM

Therefore, compared to totally unsupervised models (e.g., LDA or STM), KeyATMs seek to improve topic relevance and interpretability (by pre-determining some topics of interest), whilst allowing for some flexibility and ability to "discover" topics.

Evaluating KeyATM models

KeyATM, like other topic models, can be evaluated by using similar measures employed for LDA models (e.g., FREX scores, and semantic coherence)

Evaluating KeyATM models

KeyATM, like other topic models, can be evaluated by using similar measures employed for LDA models (e.g., FREX scores, and semantic coherence)

However, as they map texts into pre-determined categories, other types of validity assessments are possible as well (see [Ying et al., 2022])

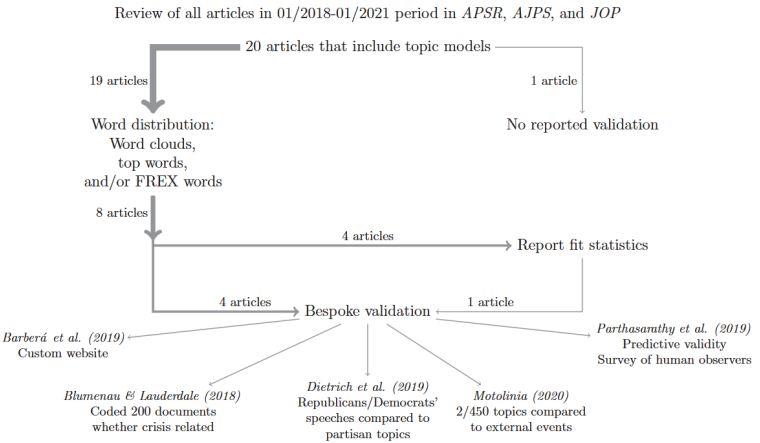


Figure 1. Survey of practices in topic model analysis in top political science journals.

Evaluating KeyATM models

"Bespoke approaches" to validate topic models [Ying et al., 2022] are tailored to the specificity of the study under consideration

However, they share the overall aim of testing a measure against substantive expectations

Evaluating KeyATM models

"Bespoke approaches" to validate topic models [Ying et al., 2022] are tailored to the specificity of the study under consideration

However, they share the overall aim of testing a measure against substantive expectations

Examples could be

- evaluate "predictive validity" of topics by checking that topics are responsive to external events
- convergent validity by showing that topics align with other measures
- use of "gold-standard" measures for comparison

Use of "gold-standard" measures

Generally, this takes the form of a comparison with human coding

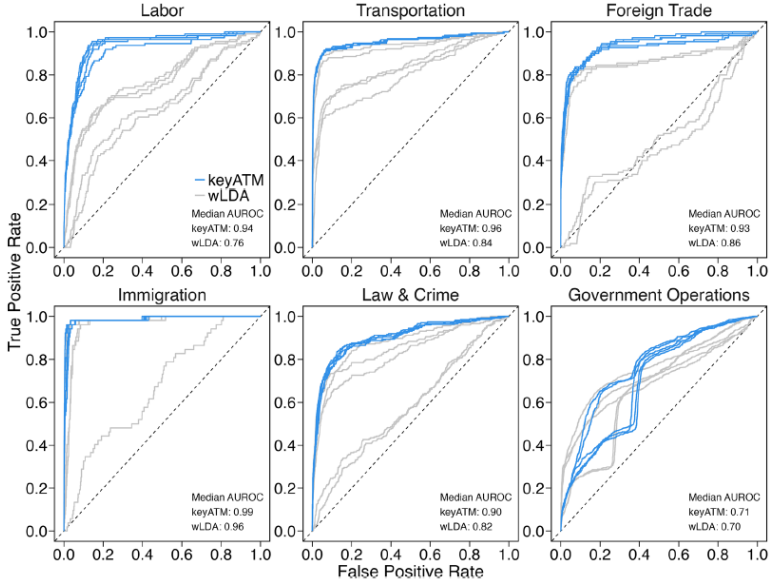
Use of "gold-standard" measures

Generally, this takes the form of a comparison with human coding

- Use of data annotated in the same categories by other scholars
- Use of data annotated by yourself/your team (see Grimmer et al. (2022, Chapter 18 "Coding a training set")
- Crowdcoded annotated data (but is still a viable option?)

Use of "gold-standard" measures

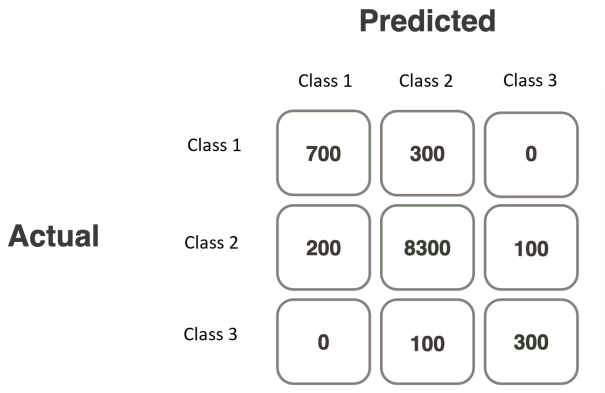
FIGURE 1 Comparison of the ROC Curves between keyATM and wLDA for Six Selected Topics



Validation with annotated data

Confusion matrix

- a *KxK* cross-tabulation of predicted classes and gold-standard classes



Validation with annotated data

Confusion matrix

- a KxK cross-tabulation of predicted classes and gold-standard classes

Validation with annotated data

Confusion matrix

- a *KxK* cross-tabulation of predicted classes and gold-standard classes

Accuracy

- The proportion of instances that have been correctly classified
- How good is my classification model, overall?

Validation with annotated data

Confusion matrix

- a *KxK* cross-tabulation of predicted classes and gold-standard classes

Accuracy

- The proportion of instances that have been correctly classified
- How good is my classification model, overall?

Precision (for category *k*)

- Number of instances *correctly* assigned to *k*, over the total number of instances assigned to *k*
- How good is my model at assigning instances to a category?

Validation with annotated data

Confusion matrix

- a *KxK* cross-tabulation of predicted classes and gold-standard classes

Accuracy

- The proportion of instances that have been correctly classified
- How good is my classification model, overall?

Precision (for category k)

- Number of instances *correctly* assigned to k, over the total number of instances assigned to k
- How good is my model at assigning instances to a category?

Recall (for category k) (sometimes called Sensitivity)

- Number of instances *correctly* assigned to k, over the total number of instances that are in category k
- How good is my model at retrieving instances from a category?

Validation with annotated data

Confusion matrix

- a *KxK* cross-tabulation of predicted classes and gold-standard classes

Accuracy

- The proportion of instances that have been correctly classified
- How good is my classification model, overall?

Precision (for category k)

- Number of instances *correctly* assigned to k, over the total number of instances assigned to k
- How good is my model at assigning instances to a category?

Recall (for category k) (sometimes called Sensitivity)

- Number of instances *correctly* assigned to k, over the total number of instances that are in category k
- How good is my model at retrieving instances from a category?

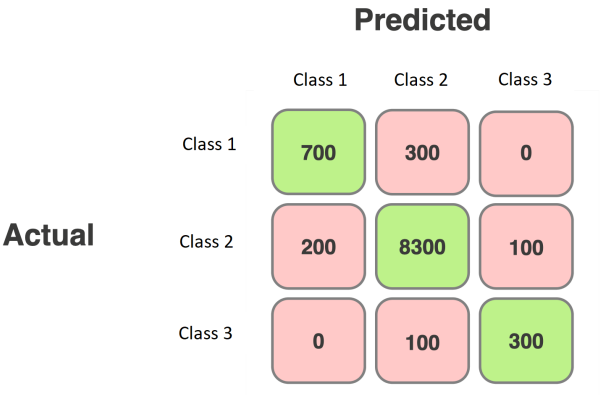
Specificity (for category k)

- Number of instances *correctly* not assigned to k, over the total number of instances that are not in category k
- How good is my model at identifying what does not belong to class k?

Validation with annotated data

Accuracy

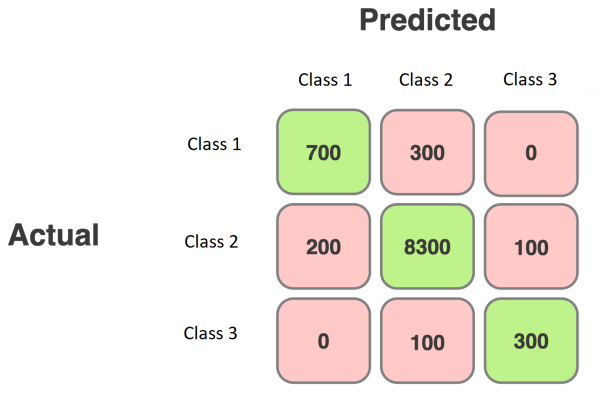
- The proportion of instances that have been correctly classified
- How good is my classification model, overall?



Validation with annotated data

Accuracy

- The proportion of instances that have been correctly classified
- How good is my classification model, overall?

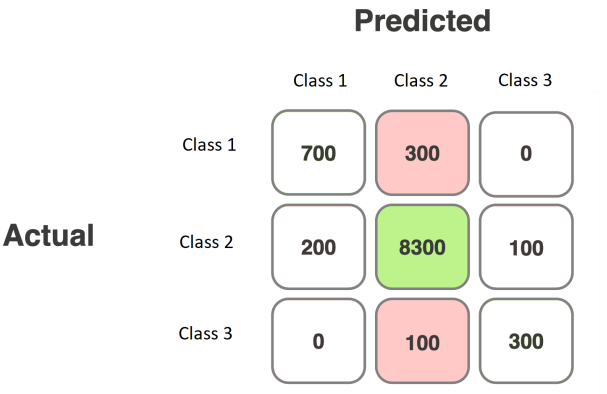


- Correct $= 700 + 8,300 + 300 = 9,300$; Total $= 10,000$
- Accuracy $= 9,300/10,000 = 0.93$

Validation with annotated data

Precision (for category k)

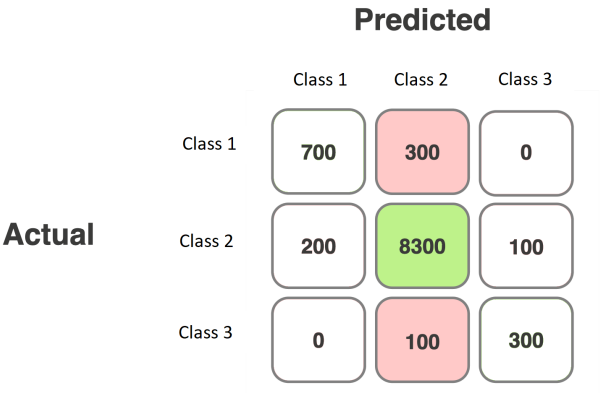
- Number of instances *correctly* assigned to k, over the total number of instances assigned to k
- How good is my model at assigning instances to a category?



Validation with annotated data

Precision (for category k)

- Number of instances *correctly* assigned to k, over the total number of instances assigned to k
- How good is my model at assigning instances to a category?

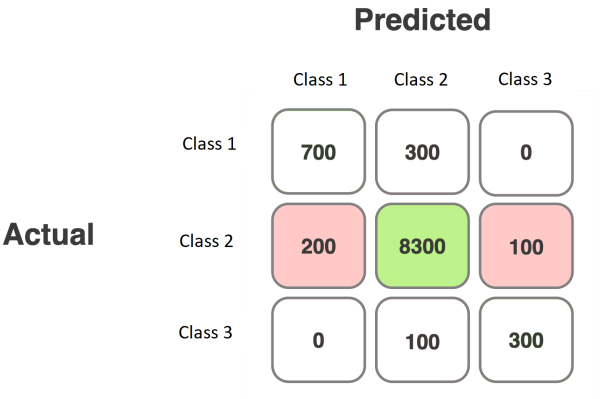


- Correct $= 8,300$; Total $= 300 + 8,300 + 100 = 8,700$
- Precision $= 8,300/8700 = 0.95$

Validation with annotated data

Recall (for category k)

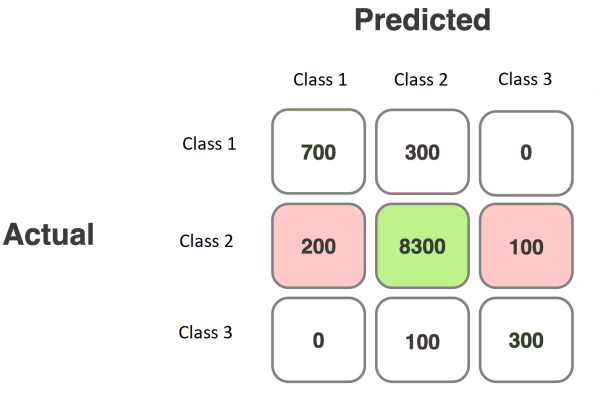
- Number of instances *correctly* assigned to k, over the total number of instances that are in category k
- How good is my model at retrieving instances from a category?



Validation with annotated data

Recall (for category k)

- Number of instances *correctly* assigned to k, over the total number of instances that are in category k
- How good is my model at retrieving instances from a category?

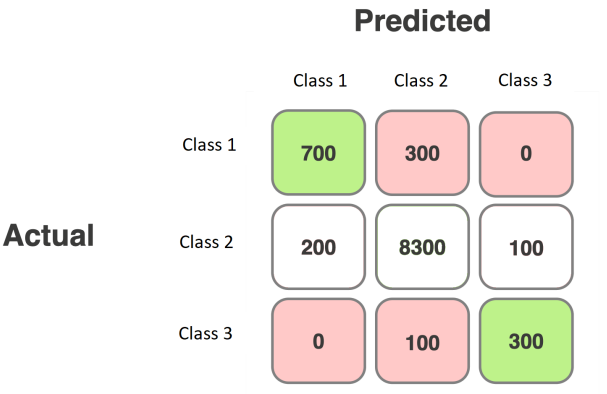


- Correct $= 8,300$; Total $= 200 + 8,300 + 100$
- Recall $= 8,300/8,600 = 0.96$

Validation with annotated data

Specificity (for category k)

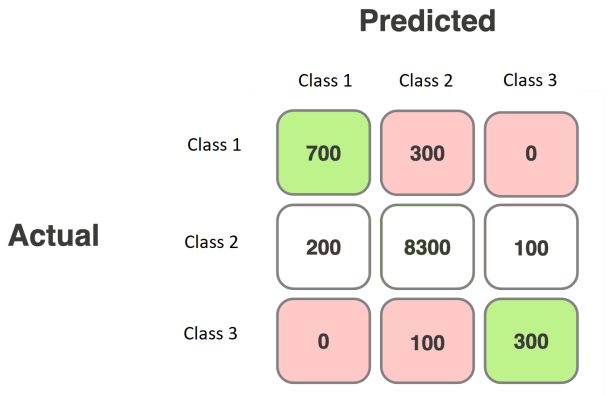
- Number of instances *correctly* not assigned to k, over the total number of instances that are not in category k
- How good is my model at identifying what does not belong to class k?



Validation with annotated data

Specificity (for category k)

- Number of instances *correctly* not assigned to k, over the total number of instances that are not in category k
- How good is my model at identifying what does not belong to class k?



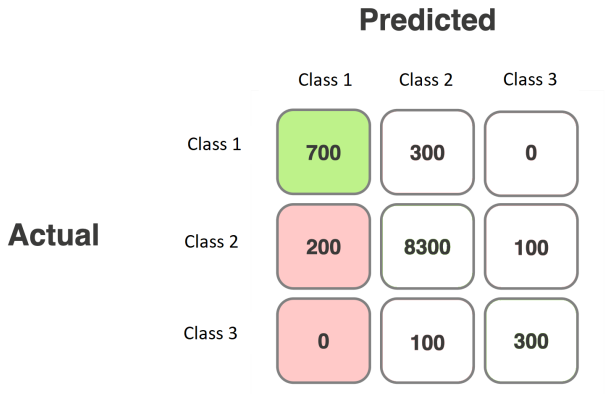
- Correct $= 700 + 300 = 1{,}000$; Total $= 700 + 300 + 100 + 300 = 1{,}400$
- Specificity $= 700/1{,}400 = 0.5$

Validation with annotated data

You can compute average precision, recall, or specificity measures by averaging them across K

Validation with annotated data

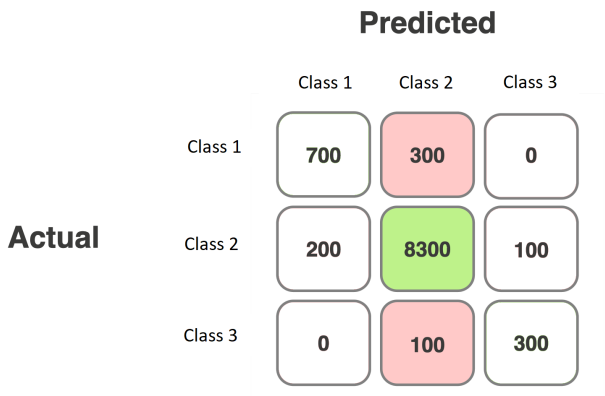
You can compute average precision, recall, or specificity measures by averaging them across K



- Precision $(k = 1) = 700/900 = 0.78$

Validation with annotated data

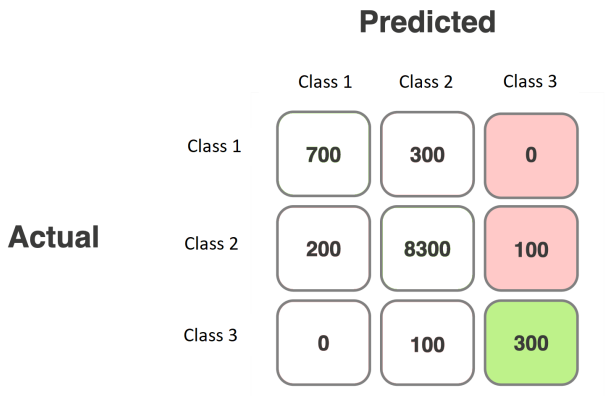
You can compute average precision, recall, or specificity measures by averaging them across K



- Precision $(k = 1) = 700/900 = 0.78$
- Precision $(k = 2) = 8,300/8,700 = 0.95$

Validation with annotated data

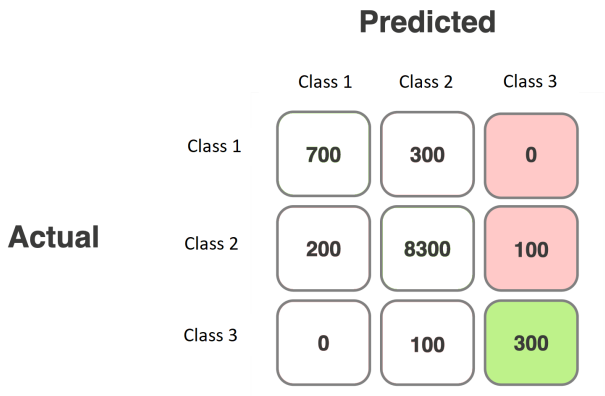
You can compute average precision, recall, or specificity measures by averaging them across K



- Precision $(k = 1) = 700/900 = 0.78$
- Precision $(k = 2) = 8,300/8,700 = 0.95$
- Precision $(k = 3) = 300/400 = 0.75$

Validation with annotated data

You can compute average precision, recall, or specificity measures by averaging them across K



- Precision $(k=1) = 700/900 = 0.78$
- Precision $(k=2) = 8,300/8,700 = 0.95$
- Precision $(k=3) = 300/400 = 0.75$
- Average Precision $= (0.78 + 0.95 + 0.75)/3 = 0.83$

We will be mostly relying on the `keyATM` package:

- `read_keywords()`
- `keyATM_read()`
- `keyATM()`
- other functions for post-estimation

We will keep using numerous `quanteda` functions. The code also contains an example of structural topic model using the `stm` package

When lost, cry for `help()`!

Supervised classification

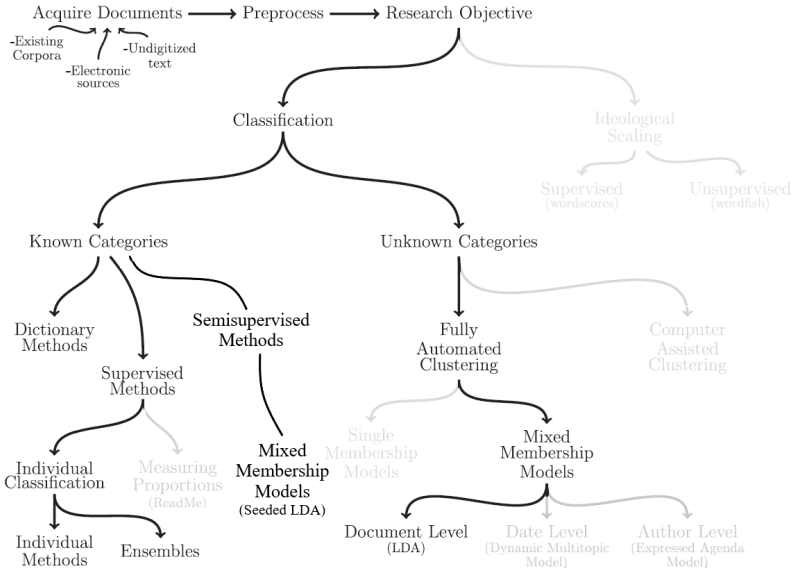


Fig. 1 An overview of text as data methods.

Supervised methods for classification

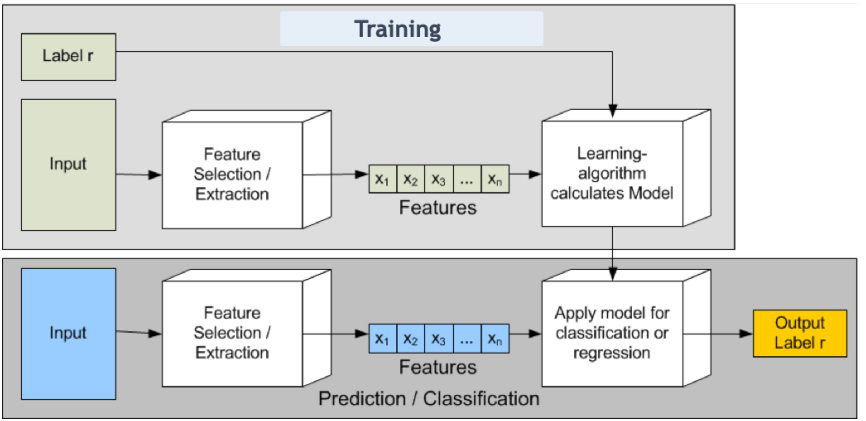
Supervised learning methods use statistical models to approximate the mapping between examples in a coded data and the labels assigned to them

Basically, they rely on examples of texts assigned to the categories of interest by coders, and "learn" the relationship between tokens in these texts and the labels assigned to them

For this reason, we say that these models are "trained" on labelled data

Supervised methods for classification

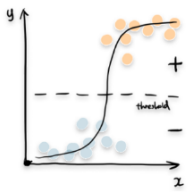
Once the model learns how tokens are mapped into labels/categories, it uses this knowledge to assign new texts to the categories based on the tokens in the new text.



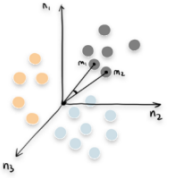
Supervised methods for classification

There are a lot of options in terms of machine learning models that can perform supervised classification

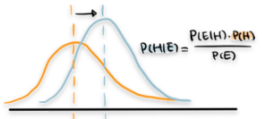
Logistic Regression



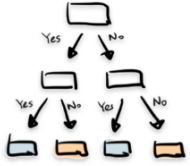
K Nearest Neighbour



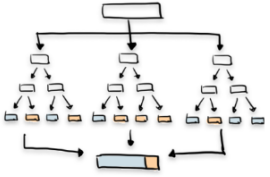
Naive Bayes



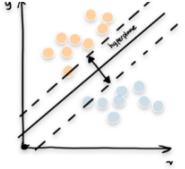
Decision Tree



Random Forest



Support Vector Machine



Supervised methods for classification

Beyond the complexity of the various algorithms, there is a core strategy:

Supervised methods for classification

Beyond the complexity of the various algorithms, there is a core strategy:

1. Define a set of categories

Supervised methods for classification

Beyond the complexity of the various algorithms, there is a core strategy:

1. Define a set of categories
2. Use those categories to label a subset of texts

Supervised methods for classification

Beyond the complexity of the various algorithms, there is a core strategy:

1. Define a set of categories
2. Use those categories to label a subset of texts
3. Train a machine learning algorithm to make predictions on new texts

Supervised methods for classification

Beyond the complexity of the various algorithms, there is a core strategy:

1. Define a set of categories
2. Use those categories to label a subset of texts
3. Train a machine learning algorithm to make predictions on new texts

As long as the machine learning algorithm predicts accurately in the kind of documents you want to analyse, it doesn't much matter what it is!

Supervised methods for classification

Beyond the complexity of the various algorithms, there is a core strategy:

1. Define a set of categories ← like in semi-supervised methods
2. Use those categories to label a subset of texts ← new!
3. Train a machine learning algorithm to make predictions on new texts ← new!

As long as the machine learning algorithm predicts accurately in the kind of documents you want to analyse, it doesn't much matter what it is!

Random forest algorithm

The random forest (RF) algorithm is a supervised method that builds on the concept of decision tree learning

Random forest algorithm

The random forest (RF) algorithm is a supervised method that builds on the concept of decision tree learning

A decision tree is a flowchart-like structure representing a decision role of the kind *if* [...] *and* [...] *then* [outcome]

Random forest algorithm

The random forest (RF) algorithm is a supervised method that builds on the concept of decision tree learning

A decision tree is a flowchart-like structure representing a decision role of the kind *if* [...] *and* [...] *then* [outcome]

- In the context of supervised text classification, you can think of the conditions as word frequencies, and the outcome as the assignment of the text to a class

Random forest algorithm

The random forest (RF) algorithm is a supervised method that builds on the concept of decision tree learning

A decision tree is a flowchart-like structure representing a decision role of the kind *if* [...] *and* [...] *then* [outcome]

- In the context of supervised text classification, you can think of the conditions as word frequencies, and the outcome as the assignment of the text to a class

Random forest algorithms "grows" many different decision trees, and then average their predictions to get the final classification

Random forest algorithm

The random forest (RF) algorithm is a supervised method that builds on the concept of decision tree learning

A decision tree is a flowchart-like structure representing a decision role of the kind *if* [...] *and* [...] *then* [outcome]

- In the context of supervised text classification, you can think of the conditions as word frequencies, and the outcome as the assignment of the text to a class

Random forest algorithms "grows" many different decision trees, and then average their predictions to get the final classification

- They are an "ensable" method

Random forest algorithm

The random forest (RF) algorithm is a supervised method that builds on the concept of decision tree learning

A decision tree is a flowchart-like structure representing a decision role of the kind *if* [...] *and* [...] *then* [outcome]

- In the context of supervised text classification, you can think of the conditions as word frequencies, and the outcome as the assignment of the text to a class

Random forest algorithms "grows" many different decision trees, and then average their predictions to get the final classification

- They are an "ensable" method
- By growing multiple and slightly different trees, RF algorithms can make more robust predictions than by relying just on one tree

Random forest algorithm

An annotated dfm is used to train an algorithm to learn how tokens map into labels.

Text	*Features/Tokens*				*Label*
	market	party	labour	budget	
doc_1	2	2	13	0	politics
doc_2	3	0	0	7	economy
doc_3	2	5	0	0	politics
doc_4	3	0	1	0	economy
doc_5	7	2	0	0	economy
$\vdots$					$\vdots$
doc_{n-4}	1	7	4	0	politics
doc_{n-3}	7	2	0	0	economy
doc_{n-2}	6	1	5	0	economy
doc_{n-1}	4	10	0	0	politics
doc_n	4	3	9	5	economy

Random forest algorithm

The training data is used in a random process to generate many decision trees, each one making a prediction

- Sample the original data to get an unique training dataset for a tree

Features/Tokens				*Label*
market	party	labour	budget	
2	2	13	0	politics
3	0	0	7	economy
2	5	0	0	politics
3	0	1	0	economy
7	2	0	0	economy
1	7	4	0	politics
7	2	0	0	economy
6	1	5	0	economy
4	10	0	0	politics
4	3	9	5	economy

Random forest algorithm

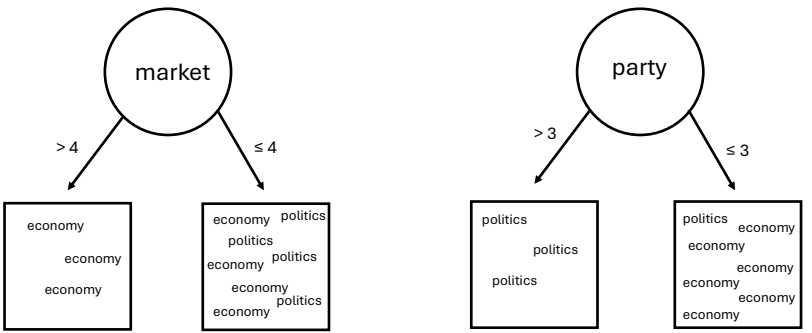
The training data is used in a random process to generate many decision trees, each one making a prediction

- Randomly sample a number of features to split the data
- Among those feature, select the one that splits the data best

Random forest algorithm

The training data is used in a random process to generate many decision trees, each one making a prediction

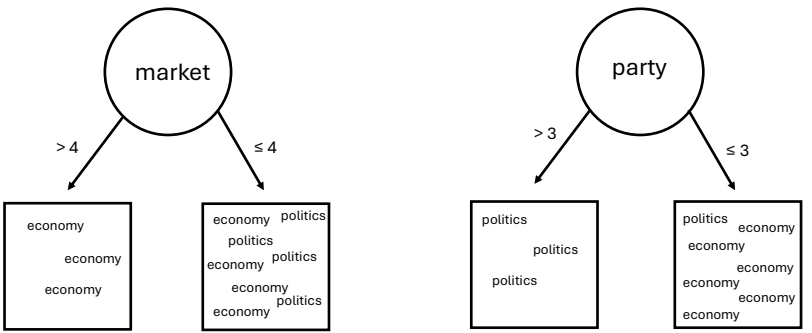
- Randomly sample a number of features to split the data
- Among those feature, select the one that splits the data best



Random forest algorithm

The training data is used in a random process to generate many decision trees, each one making a prediction

- Randomly sample a number of features to split the data
- Among those feature, select the one that splits the data best

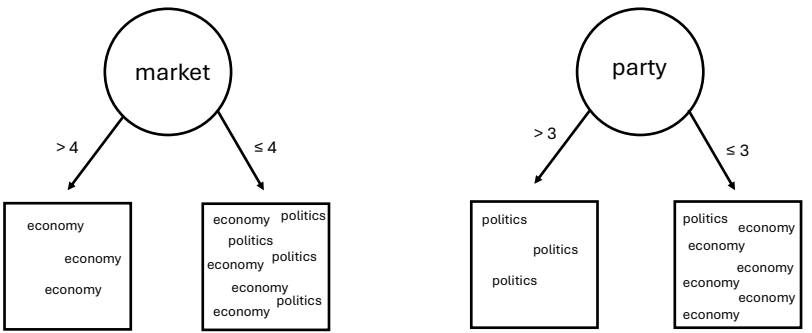


- Compare nodes "purity" using a measure called *entropy*, which basically tells us how much information is needed in a node to classify an object with certainty

Random forest algorithm

The training data is used in a random process to generate many decision trees, each one making a prediction

- Randomly sample a number of features to split the data
- Among those feature, select the one that splits the data best



- Compare nodes "purity" using a measure called *entropy*, which basically tells us how much information is needed in a node to classify an object with certainty
- Select feature leading to nodes with lowest entropy, and split

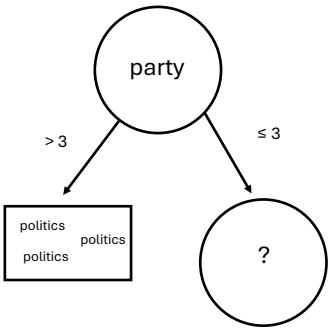
Random forest algorithm

Go ahead with further splits until the tree is fully constructed

Random forest algorithm

Go ahead with further splits until the tree is fully constructed

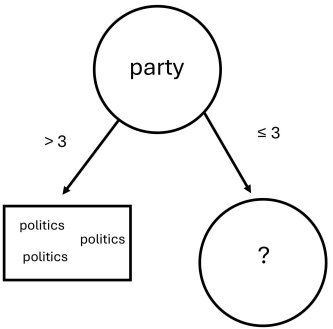
- Do the same assessment of features to split the data further

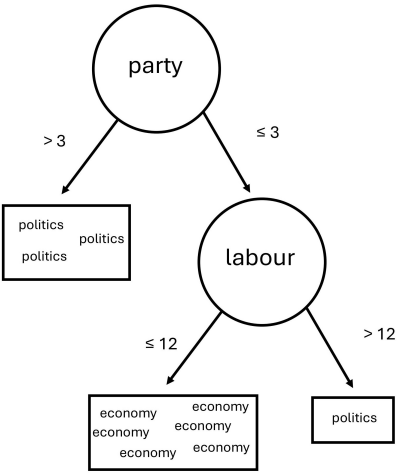


Random forest algorithm

Go ahead with further splits until the tree is fully constructed

- Do the same assessment of features to split the data further





Random forest algorithm

Once the decision tree is "fully grown", it has learned how tokens maps into labels

It can now be used to make predictions on new texts

Random forest algorithm

Once the decision tree is "fully grown", it has learned how tokens maps into labels

It can now be used to make predictions on new texts

- Imagine we want to label a text containing the tokens "market", "party", "labour", and "budget" appearing 1, 3, 9, and 0 times.

Random forest algorithm

Once the decision tree is "fully grown", it has learned how tokens maps into labels

It can now be used to make predictions on new texts

- Imagine we want to label a text containing the tokens "market", "party", "labour", and "budget" appearing 1, 3, 9, and 0 times.
- The tree has learned that *if* "party" < 3 *and* "labour" < 12 *then* $\rightarrow$ "economy"

Random forest algorithm

Once the decision tree is "fully grown", it has learned how tokens maps into labels

It can now be used to make predictions on new texts

- Imagine we want to label a text containing the tokens "market", "party", "labour", and "budget" appearing 1, 3, 9, and 0 times.
- The tree has learned that *if* "party" < 3 *and* "labour" < 12 *then* $\rightarrow$ "economy"

In a RF model, many different such trees are grown

Random forest algorithm

Once the decision tree is "fully grown", it has learned how tokens maps into labels

It can now be used to make predictions on new texts

- Imagine we want to label a text containing the tokens "market", "party", "labour", and "budget" appearing 1, 3, 9, and 0 times.
- The tree has learned that *if* "party" < 3 *and* "labour" < 12 *then* $\rightarrow$ "economy"

In a RF model, many different such trees are grown

Each one makes its own prediction for a new text

Random forest algorithm

Once the decision tree is "fully grown", it has learned how tokens maps into labels

It can now be used to make predictions on new texts

- Imagine we want to label a text containing the tokens "market", "party", "labour", and "budget" appearing 1, 3, 9, and 0 times.
- The tree has learned that *if* "party" < 3 *and* "labour" < 12 *then* $\rightarrow$ "economy"

In a RF model, many different such trees are grown

Each one makes its own prediction for a new text

Then, the label "voted" by most trees is the final model prediction

How to grow a random forest

The structure and behaviour of machine learning models is governed by quantities called “parameters”

How to grow a random forest

The structure and behaviour of machine learning models is governed by quantities called “parameters”

Each type of model has its own set of parameters

How to grow a random forest

The structure and behaviour of machine learning models is governed by quantities called "parameters"

Each type of model has its own set of parameters

In the case of random forest algorithms, the main parameters are those determining:

- how many trees are there?
- how many features to evaluate at each split?
- when to stop growing a tree?

How to grow a random forest

The structure and behaviour of machine learning models is governed by quantities called "parameters"

Each type of model has its own set of parameters

In the case of random forest algorithms, the main parameters are those determining:

- how many trees are there?
- how many features to evaluate at each split?
- when to stop growing a tree?

Experience may provide us with "rules of thumb" on acceptable parameter values

Hyperparameter tuning is useful in finding optimal parameter values

How to grow a random forest

The structure and behaviour of machine learning models is governed by quantities called "parameters"

Each type of model has its own set of parameters

In the case of random forest algorithms, the main parameters are those determining:

- how many trees are there?
- how many features to evaluate at each split?
- when to stop growing a tree?

Experience may provide us with "rules of thumb" on acceptable parameter values

Hyperparameter tuning is useful in finding optimal parameter values

Model tuning $\neq$ Model training

Tuning a machine learning algorithm

Tuning means finding optimal parameter values for training the model

Tuning a machine learning algorithm

Tuning means finding optimal parameter values for training the model

1. Decide which parameters can/needs to be tuned (this depends on the model and on the importance of the parameter for classification performance)

Tuning a machine learning algorithm

Tuning means finding optimal parameter values for training the model

1. Decide which parameters can/needs to be tuned (this depends on the model and on the importance of the parameter for classification performance)
2. Select the possible parameter values that you want to try for each parameter

Tuning a machine learning algorithm

Tuning means finding optimal parameter values for training the model

1. Decide which parameters can/needs to be tuned (this depends on the model and on the importance of the parameter for classification performance)
2. Select the possible parameter values that you want to try for each parameter
3. Split your data in train, validation and test set (alternatively, you can use cross-validation during tuning. It is important that all final evaluations are made on the test set)

Tuning a machine learning algorithm

Tuning means finding optimal parameter values for training the model

1. Decide which parameters can/needs to be tuned (this depends on the model and on the importance of the parameter for classification performance)
2. Select the possible parameter values that you want to try for each parameter
3. Split your data in train, validation and test set (alternatively, you can use cross-validation during tuning. It is important that all final evaluations are made on the test set)
4. Estimate different models, one for each combination of parameter values, and assess their performance

Tuning a machine learning algorithm

Tuning means finding optimal parameter values for training the model

1. Decide which parameters can/needs to be tuned (this depends on the model and on the importance of the parameter for classification performance)
2. Select the possible parameter values that you want to try for each parameter
3. Split your data in train, validation and test set (alternatively, you can use cross-validation during tuning. It is important that all final evaluations are made on the test set)
4. Estimate different models, one for each combination of parameter values, and assess their performance
   - 4.1 Fit the model on the training data
   - 4.2 Assess model performance on a validation set
   - 4.3 Select parameter values used for the best performing model

Tuning a machine learning algorithm

Tuning means finding optimal parameter values for training the model

1. Decide which parameters can/needs to be tuned (this depends on the model and on the importance of the parameter for classification performance)
2. Select the possible parameter values that you want to try for each parameter
3. Split your data in train, validation and test set (alternatively, you can use cross-validation during tuning. It is important that all final evaluations are made on the test set)
4. Estimate different models, one for each combination of parameter values, and assess their performance
   - 4.1 Fit the model on the training data
   - 4.2 Assess model performance on a validation set
   - 4.3 Select parameter values used for the best performing model
5. Test model on test set to simulate real world scenario (validation set influences the model as it is used for tuning, hence it is not anymore a credible benchmark)

Tuning a machine learning algorithm

Tuning means finding optimal parameter values for training the model

1. Decide which parameters can/needs to be tuned (this depends on the model and on the importance of the parameter for classification performance)
2. Select the possible parameter values that you want to try for each parameter
3. Split your data in train, validation and test set (alternatively, you can use cross-validation during tuning. It is important that all final evaluations are made on the test set)
4. Estimate different models, one for each combination of parameter values, and assess their performance
   - 4.1 Fit the model on the training data
   - 4.2 Assess model performance on a validation set
   - 4.3 Select parameter values used for the best performing model
5. Test model on test set to simulate real world scenario (validation set influences the model as it is used for tuning, hence it is not anymore a credible benchmark)
6. If test performance is satisfactory and no additional tuning is needed, you can train the model once again on the full labelled data (with the parameter values decided after tuning) and use it to label unseen data

Validation

Validation in the context of supervised learning

In supervised classification tasks, performing a train-test split is essential to evaluate the model's ability to generalize to unseen data

Validation in the context of supervised learning

In supervised classification tasks, performing a train-test split is essential to evaluate the model's ability to generalize to unseen data

The dataset is divided into a training set (used to fit the model) and a test set (used to assess performance)

Validation in the context of supervised learning

In supervised classification tasks, performing a train-test split is essential to evaluate the model's ability to generalize to unseen data

The dataset is divided into a training set (used to fit the model) and a test set (used to assess performance)

Model tuning might require a further split in a validation set, or techniques like k-fold cross-validation

Validation in the context of supervised learning

In supervised classification tasks, performing a train-test split is essential to evaluate the model's ability to generalize to unseen data

The dataset is divided into a training set (used to fit the model) and a test set (used to assess performance)

Model tuning might require a further split in a validation set, or techniques like k-fold cross-validation

All this is to avoid "data leakage" and ensure that the model performs well not only on the training data but also on new, real-world examples (avoiding "over-fitting")

(More on) validation with annotated data

Confusion matrix

- a *KxK* cross-tabulation of predicted classes and gold-standard classes

Accuracy

- The proportion of instances that have been correctly classified
- How good is my classification model, overall?

Precision (for category k)

- Number of instances *correctly* assigned to k, over the total number of instances assigned to k
- How good is my model at assigning instances to a category?

Recall (for category k) (sometimes called Sensitivity)

- Number of instances *correctly* assigned to k, over the total number of instances that are in category k
- How good is my model at retrieving instances from a category?

Specificity (for category k)

- Number of instances *correctly* not assigned to k, over the total number of instances that are not in category k
- How good is my model at identifying what does not belong to class k?

(More on) validation with annotated data

In some cases the accuracy measure is too basic and can give us some biased evaluation of classification performance

Accuracy can be problematic in case of multi-class classification tasks, and definitely to avoid in case of class imbalance

(More on) validation with annotated data

In some cases the accuracy measure is too basic and can give us some biased evaluation of classification performance

Accuracy can be problematic in case of multi-class classification tasks, and definitely to avoid in case of class imbalance

Other measures are more robust and, therefore, preferable. Most of them try to combine the information coming from precision and recall metrics into one score

(More on) validation with annotated data

In some cases the accuracy measure is too basic and can give us some biased evaluation of classification performance

Accuracy can be problematic in case of multi-class classification tasks, and definitely to avoid in case of class imbalance

Other measures are more robust and, therefore, preferable. Most of them try to combine the information coming from precision and recall metrics into one score

Balanced Accuracy

- It is essentially the average of recall across classes
- It tend to converge to the accuracy measure if the dataset is balanced

(More on) validation with annotated data

In some cases the accuracy measure is too basic and can give us some biased evaluation of classification performance

Accuracy can be problematic in case of multi-class classification tasks, and definitely to avoid in case of class imbalance

Other measures are more robust and, therefore, preferable. Most of them try to combine the information coming from precision and recall metrics into one score

Balanced Accuracy

- It is essentially the average of recall across classes
- It tend to converge to the accuracy measure if the dataset is balanced

F1-Score

- It is the harmonic mean of precision and recall
- Hence, it is good to spot weak points of a prediction algorithm

(More on) validation with annotated data

In some cases the accuracy measure is too basic and can give us some biased evaluation of classification performance

Accuracy can be problematic in case of multi-class classification tasks, and definitely to avoid in case of class imbalance

Other measures are more robust and, therefore, preferable. Most of them try to combine the information coming from precision and recall metrics into one score

Balanced Accuracy

- It is essentially the average of recall across classes
- It tend to converge to the accuracy measure if the dataset is balanced

F1-Score

- It is the harmonic mean of precision and recall
- Hence, it is good to spot weak points of a prediction algorithm

- Grandini et al. 2020 provide a super useful review of such measures (see reading list)

We will be mostly relying on the `randomForest` package:

- `randomForest()`
- other functions for post-estimation and validation

Other useful packages for machine learning classifiers are: `class`, `naivebayes`,, and `xgboost`

When lost, cry for `help()`!

Recap

- Depending on the amount of labelled data used to train our models, we talk of unsupervised, semisupervised or supervised methods
- Unsupervised methods (e.g., LDA), require little ex ante work, but more effort in terms of ex post interpretation
- Semisupervised and supervised methods are preferable when we are interested in pre-determined labels or categories
- They require different amount of annotated input data, but can be also more easily assessed against gold standard measures
- Validation is essential for all models, and many metrics are available to summarise the performance of a classifier

Next session:

Word-embeddings approaches
and large language models

References I

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003).
Latent dirichlet allocation.
Journal of machine Learning research, 3(Jan):993–1022.

Eshima, S., Imai, K., and Sasaki, T. (2024).
Keyword-assisted topic models.
American Journal of Political Science, 68(2):730–750.

Maier, D., Waldherr, A., Miltner, P., Wiedemann, G., Niekler, A., Keinert, A., Pfetsch, B., Heyer, G., Reber, U., Häussler, T., et al. (2018).
Applying lda topic modeling in communication research: Toward a valid and reliable methodology.
Communication Methods and Measures, 12(2-3):13–38.

Moschella, M., Pinto, L., and Martocchia Diodati, N. (2020).
Let's speak more? how the ecb responds to public contestation.
Journal of European public policy, 27(3):400–418.

Roberts, M. E., Stewart, B. M., and Airoldi, E. M. (2016).
A model of text for experimentation in the social sciences.
Journal of the American Statistical Association, 111(515):988–1003.

References II


Roberts, M. E., Stewart, B. M., Tingley, D., Lucas, C., Leder-Luis, J., Gadarian, S. K., Albertson, B., and Rand, D. G. (2014).
Structural topic models for open-ended survey responses.
American journal of political science, 58(4):1064–1082.


Watanabe, K. and Zhou, Y. (2020).
Theory-driven analysis of large corpora: Semisupervised topic classification of the UN speeches.
Social Science Computer Review, pages 1–21.


Ying, L., Montgomery, J. M., and Stewart, B. M. (2022).
Topics, concepts, and measurement: A crowdsourced procedure for validating topics as measures.
Political Analysis, 30(4):570–589.