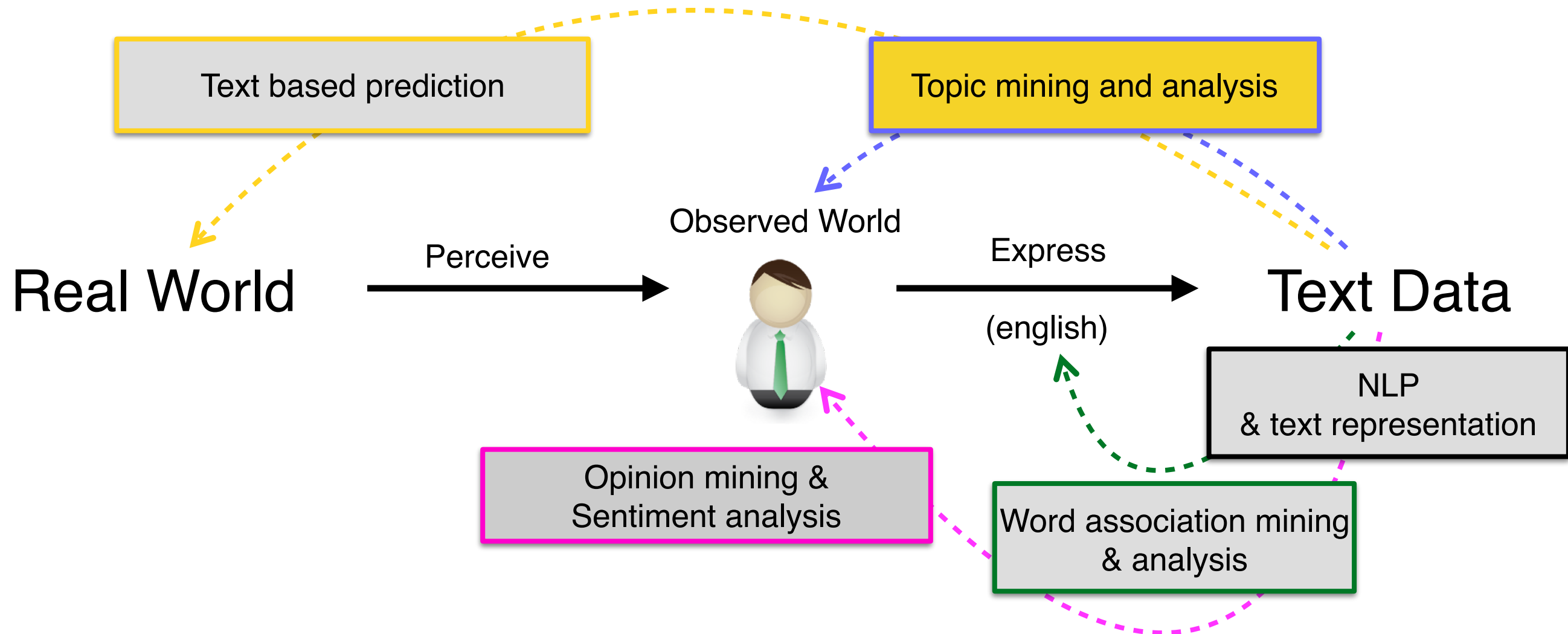


Web Content Mining

Topic Modeling and Text Classification

— Session 2 —

Landscape of Text Mining and Analytics

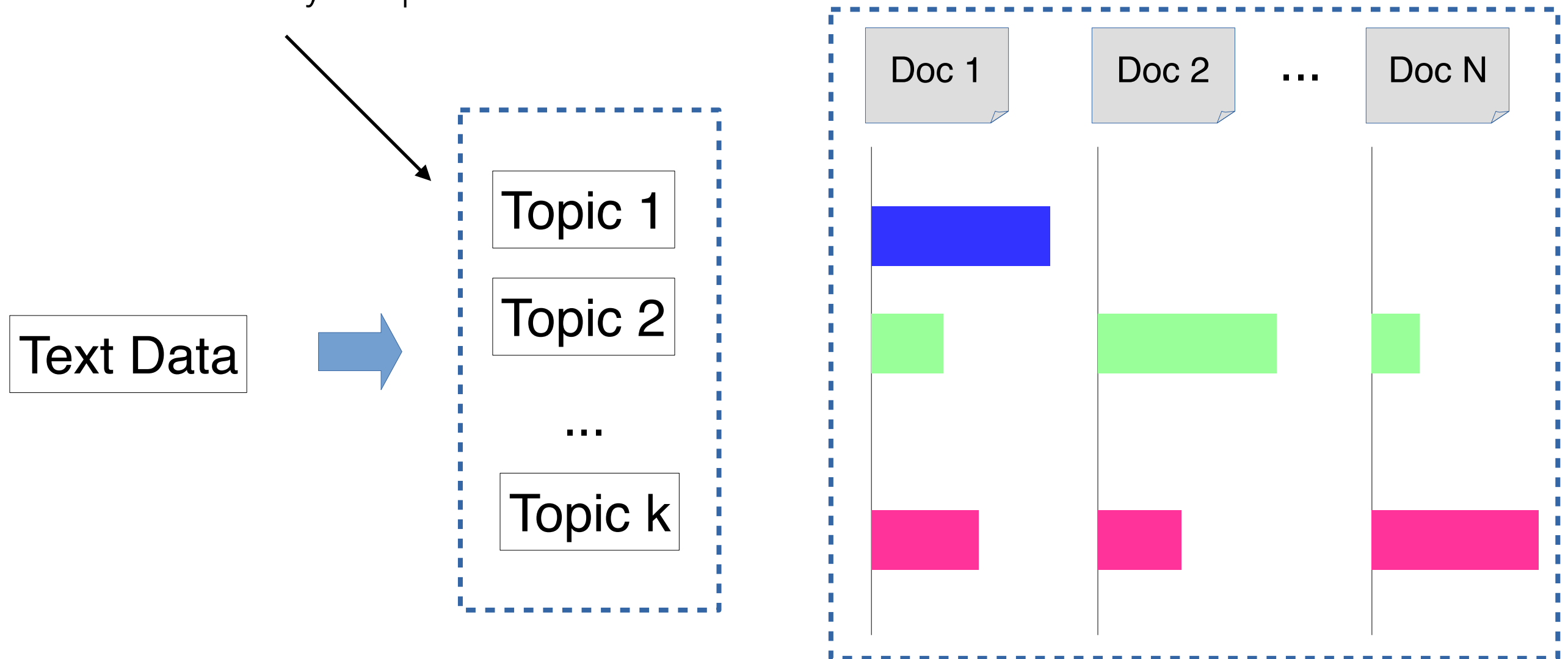


Topic Mining and Analysis: Motivation

- ▶ Topic~ main idea discussed in text data
 - ▶ Theme/subject of a discussion or conversation
 - ▶ Different granularities (e.g., topic of a sentence, an article, etc.)
- ▶ Many application require discovery of topics in text
 - ▶ What are Twitter users talking about today?
 - ▶ What are the current research topics in data mining? How are they different from those 5 years ago?
 - ▶ What do people like about the iPhone X? What do they dislike?
 - ▶ What were the major topics debated in 2016 presidential election?

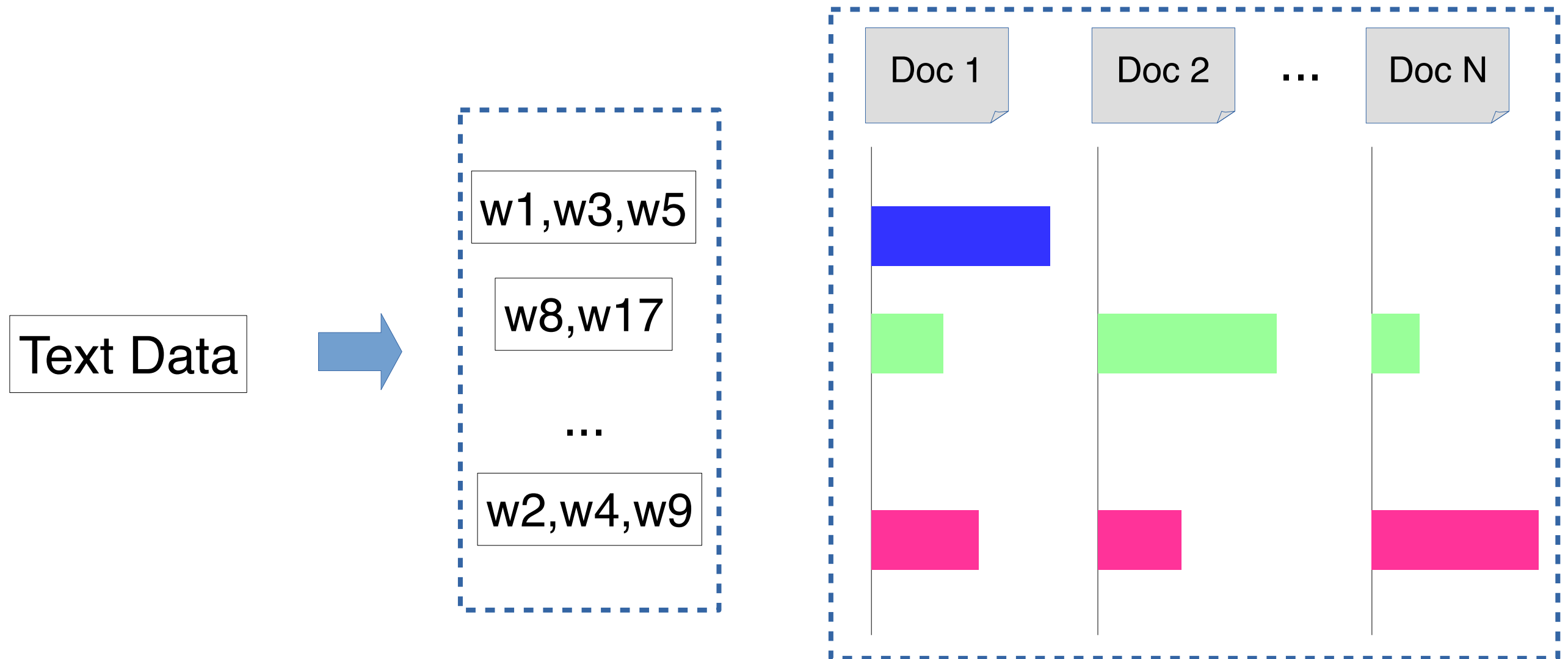
Task of Topic Mining and Analysis

Task 1: Discovery k topics



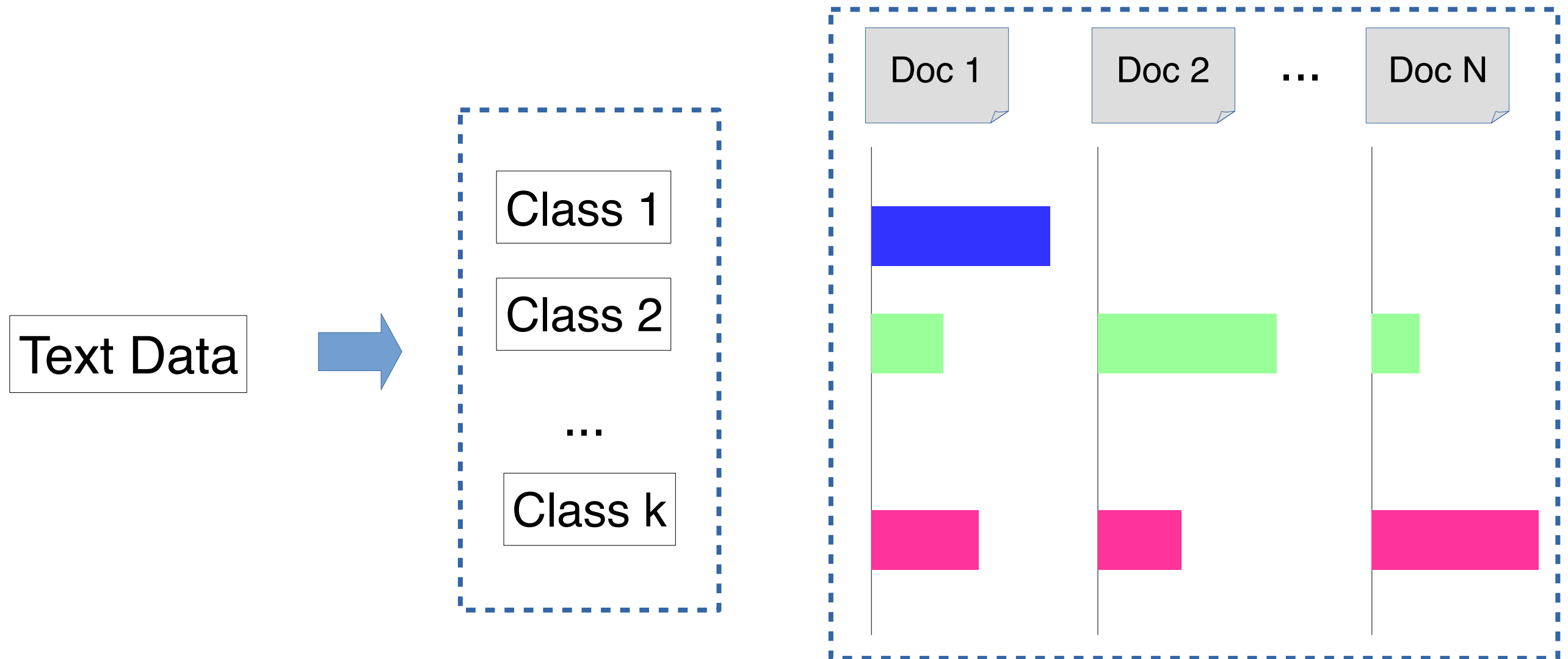
Task 2: Figure out which documents cover which topics

Task of Topic Mining and Analysis



Unsupervised problem
(clustering)

Task of Topic Mining and Analysis



Supervised problem
(classification)

Topic mining and Analysis

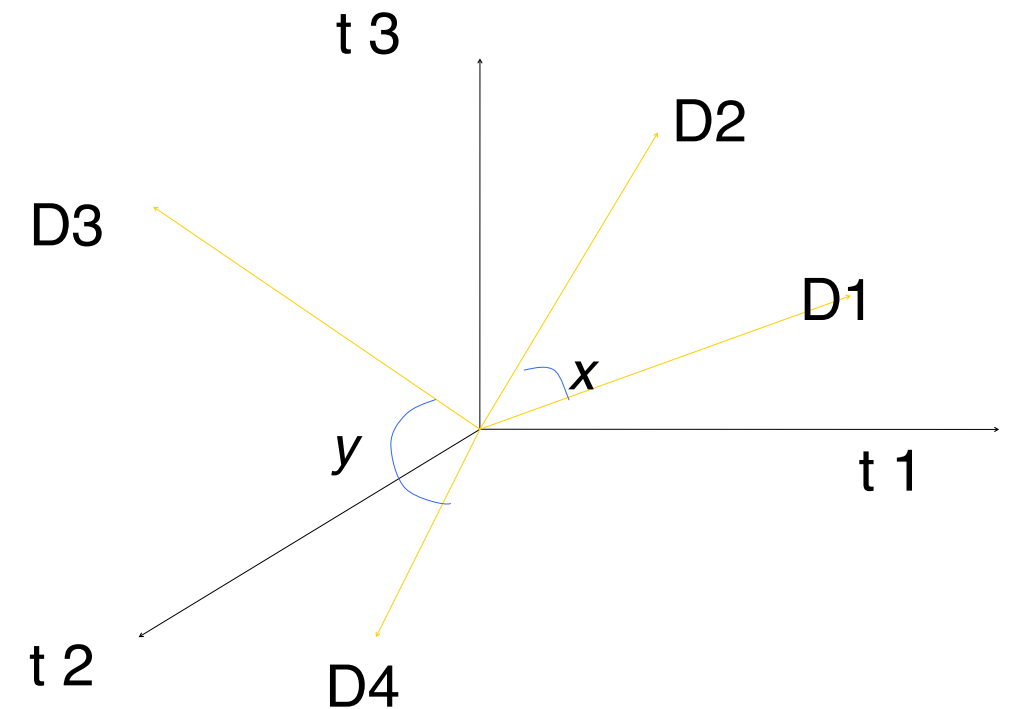
- **Group of documents (unsupervised)**
 - **Clustering**
- Group of documents but into classes (supervised)
 - Classification
- Group of words - dimensionality reduction (We are going to see them in the unsupervised part)
 - NNMF, LDA, PLSA, etc.

Clustering

- ▶ Clustering of documents is a way to discovery related documents and topics
- ▶ It is the process of grouping a set of objects into classes of similar objects
 - ▶ Documents within a cluster should be similar.
 - ▶ Documents from different clusters should be dissimilar
- ▶ Many existing algorithms
 - ▶ K-means, E.M., etc.
- ▶ A representation for each document is demanded (or document similarity)
 - ▶ Term-document or document-term matrix
 - ▶ The frequency is frequently used, but many others weighted models are available (tf-idf). Their use depends of the chosen algorithm

Issues for clustering

- ▶ Representation for clustering
 - ▶ Document representation
 - ▶ Vector space? Normalization?
 - ▶ Centroids aren't length normalized
 - ▶ Need a notion of similarity/distance
- ▶ How many clusters?
 - ▶ Fixed a priori?
 - ▶ Completely data driven?
 - Avoid “trivial” clusters - too large or small



Document Representation

TF-IDF

- ▶ Frequent terms are less informative than rare terms
- ▶ A document containing a frequent term is more likely to be relevant than a document that doesn't
- ▶ But it's not a sure indicator of relevance..

$$\text{TF-IDF} \Rightarrow w_{t,d} = \underbrace{\log(1 + \text{tf}_{t,d})}_{\text{TF}} \times \underbrace{\log_{10}(N / \text{df}_t)}_{\text{IDF}}$$

- ▶ Increases with the number of occurrences within a document
- ▶ Increases with the rarity of the term in the collection

Document Representation

Term-Document Matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
anthony	1	1	0	0	0	1
brutus	1	1	0	1	0	0
caesar	1	1	0	1	1	1
calpurnia	0	1	0	0	0	0
cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Binary matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
anthony	5.25	3.18	0.0	0.0	0.0	0.35
brutus	1.21	6.10	0.0	1.0	0.0	0.0
caesar	8.59	2.54	0.0	1.51	0.25	0.0
calpurnia	0.0	1.54	0.0	0.0	0.0	0.0
cleopatra	2.85	0.0	0.0	0.0	0.0	0.0
mercy	1.51	0.0	1.90	0.12	5.25	0.88
worser	1.37	0.0	0.11	4.15	0.25	1.95

TF-IDF

Values used in these matrices are known as the weighting schema. They alter the obtained results (clustering or classification). Often the frequency is used, but other options are also available:

Term frequency		Document frequency		Normalization	
n (natural)	$tf_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(tf_{t,d})$	t (idf)	$\log \frac{N}{df_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2 + w_2^2 + \dots + w_M^2}}$
a (augmented)	$0.5 + \frac{0.5 \times tf_{t,d}}{\max_t(tf_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N - df_t}{df_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases} 1 & \text{if } tf_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/CharLength^\alpha, \alpha < 1$
L (log ave)	$\frac{1 + \log(tf_{t,d})}{1 + \log(\text{ave}_{t \in d}(tf_{t,d}))}$				

Why frequency is not the best option? Document size strongly affects similarity values!!!

Documents as vectors

- ▶ So we have a $|V|$ -dimensional vector space
- ▶ Terms are axes of the space
- ▶ Documents are points or vectors in this space
- ▶ Very high-dimensional: millions of dimensions when you apply this to big collections
- ▶ These are very sparse vectors - most entries are zero

Notion of similarity/distance

- ▶ Ideal: semantic similarity (in your dreams)
- ▶ Practical: term-statistical similarity
 - ▶ Cosine similarity
 - ▶ Docs as vectors
- ▶ For many algorithms, easier to think in terms of a distance (rather than similarity) between docs
- ▶ It is easier to talk about Euclidean distance, but real implementations use cosine similarity

Hard vs. soft clustering

- ▶ Hard clustering: Each document belongs to exactly one cluster
 - ▶ More common and easier to do
- ▶ Soft clustering: A document can belong to more than one cluster.
 - ▶ Makes more sense for applications like creating browsable hierarchies
 - ▶ You may want to put a pair of sneakers in two clusters: (i) sports apparel and (ii) shoes
 - ▶ You can only do that with a soft clustering approach.

What Is a Good Clustering?

- ▶ Internal criterion: A good clustering will produce high quality clusters in which:
 - ▶ the intra-class (that is, intra-cluster) similarity is high
 - ▶ the inter-class similarity is low
 - ▶ The measured quality of a clustering depends on both the document representation and the similarity measure used
- ▶ External criterion: The quality of a clustering is also measured by its ability to discover some or all of the hidden patterns or latent classes
 - ▶ Assessable with gold standard data

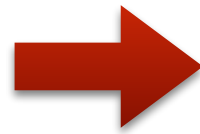
Clustering algorithms

- ▶ Hard clustering

- ▶ K-means

- ▶ Hard EM

- ▶ Hierarchical clustering



Not seen in
this course

- ▶ Soft clustering

- ▶ Latent Semantic Analysis (LSA)

- ▶ Non-negative matrix factorization

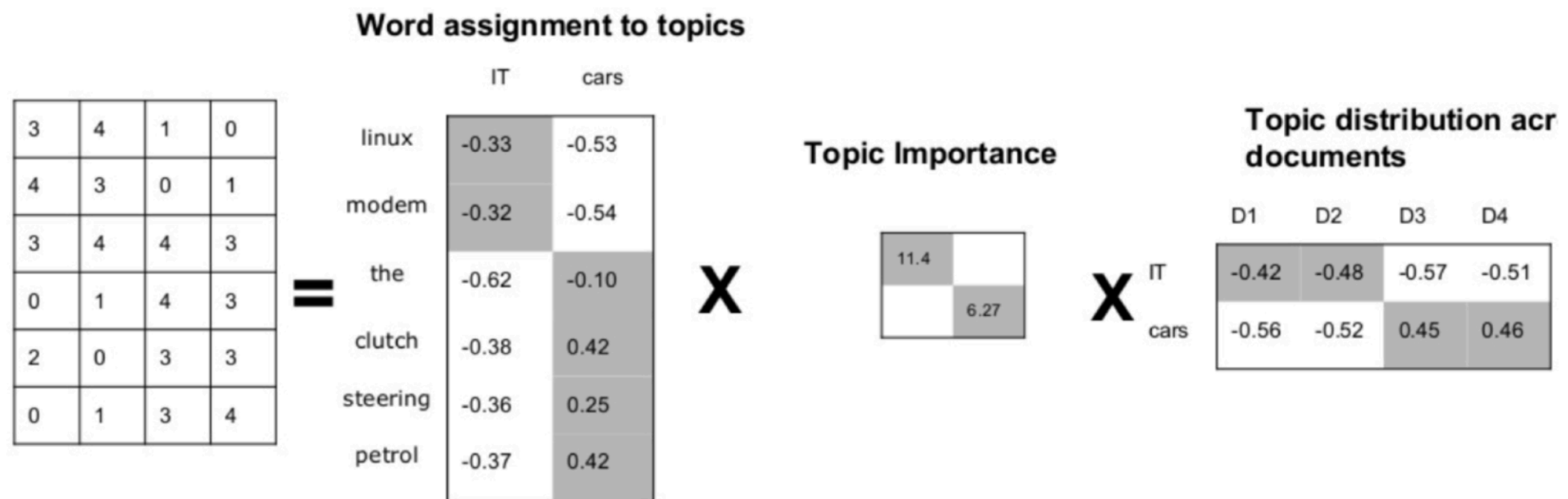
- ▶ Latent Dirichlet allocation

- ▶ Many others but these are the main used in text clustering

LSA

Latent Semantic Analysis

- ▶ Proposed by Deerwester et al. (JASIST, 1990)
- ▶ Perform a SVD to the term-document matrix
- ▶ Result: low-rank approximation

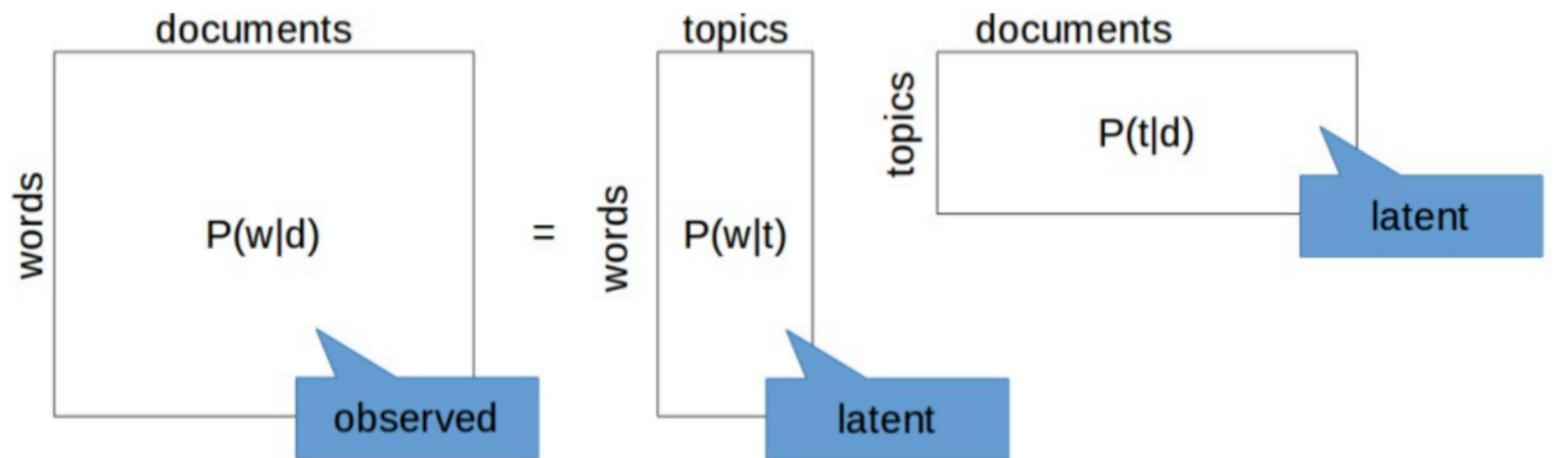


Problem: how can negative weights be interpreted ?

NMF and pLSA

Two alternatives

- ▶ NMF (Paatero et Tapper, 1990). Matrix decomposition with constraint on the positivity of the resulting matrices
- ▶ pLSA (Hofmann, 1999). Probabilistic version of LSA. Weights are probabilities (positive and interpretable)

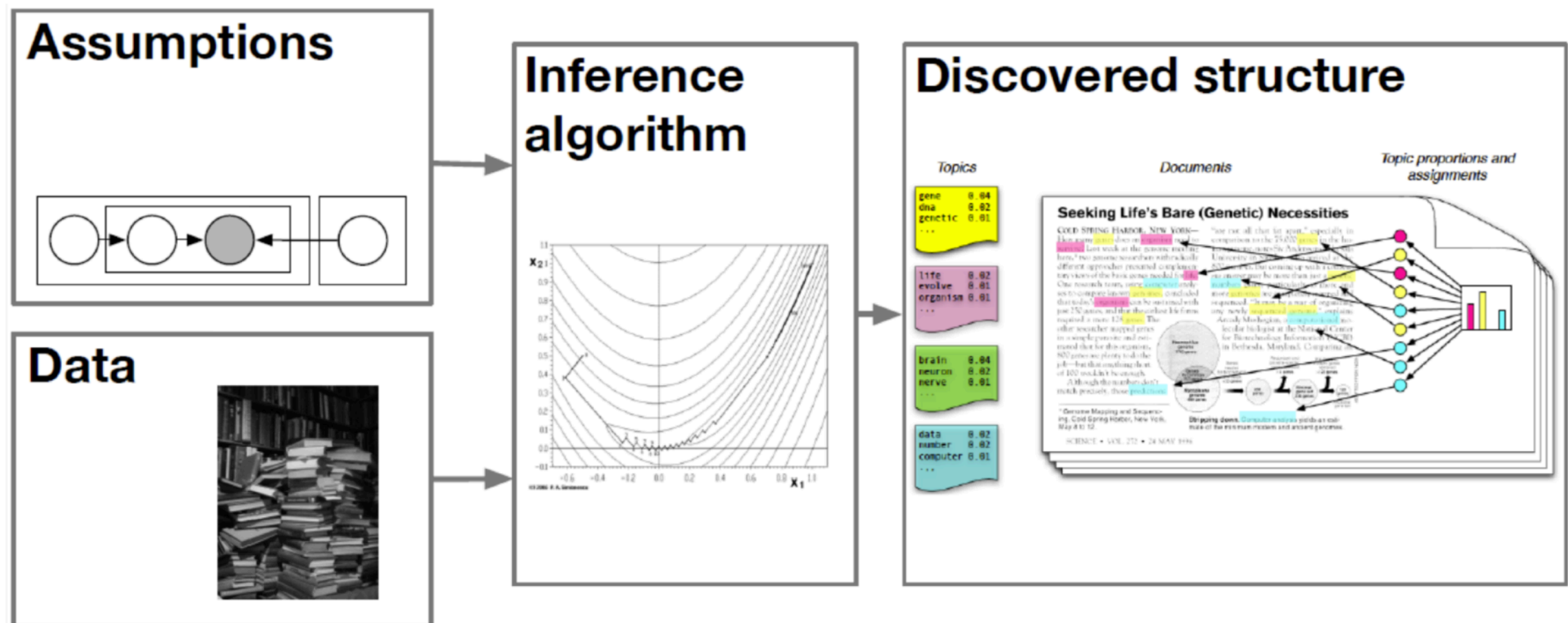


Problem: overfitting and model not really generative (impossible to apply it on new documents)

LDA

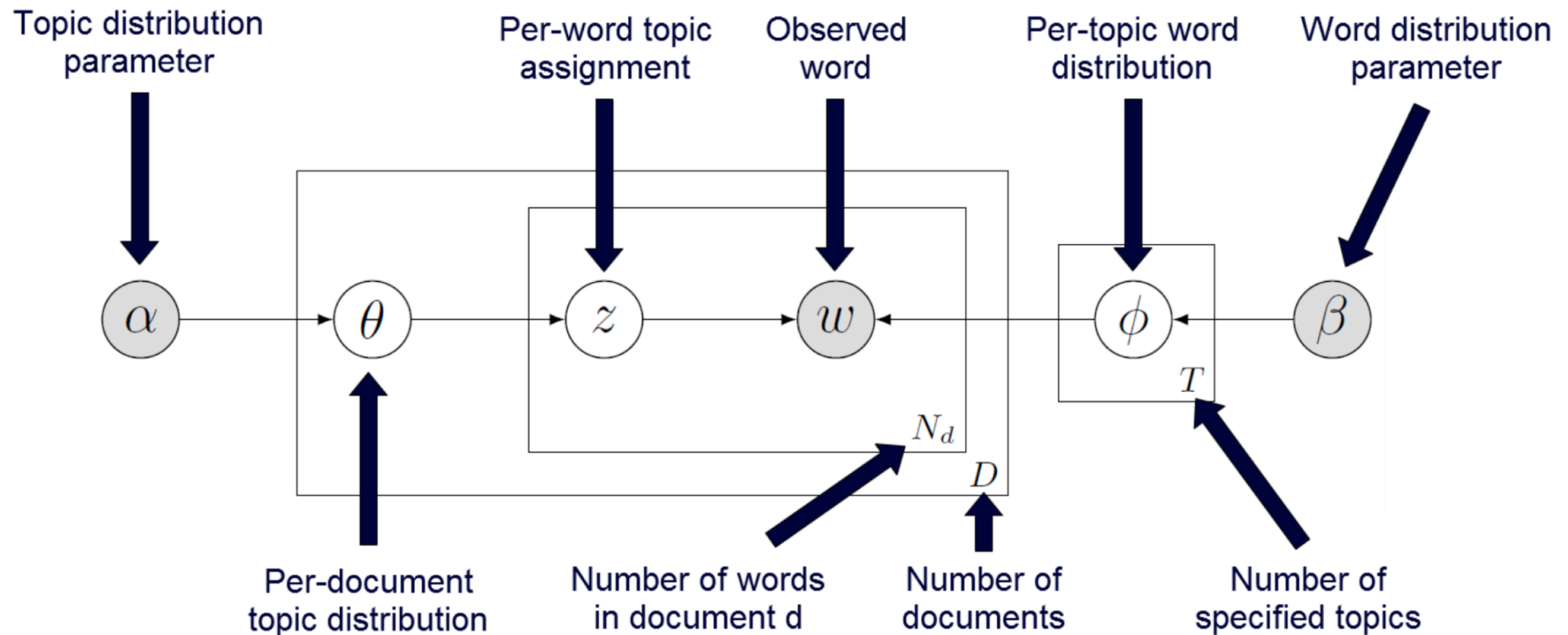
Latent Dirichlet Allocation

- ▶ Blei et al. (2001)
- ▶ Fully generative model
- ▶ Inspired by probabilistic graphic model



LDA

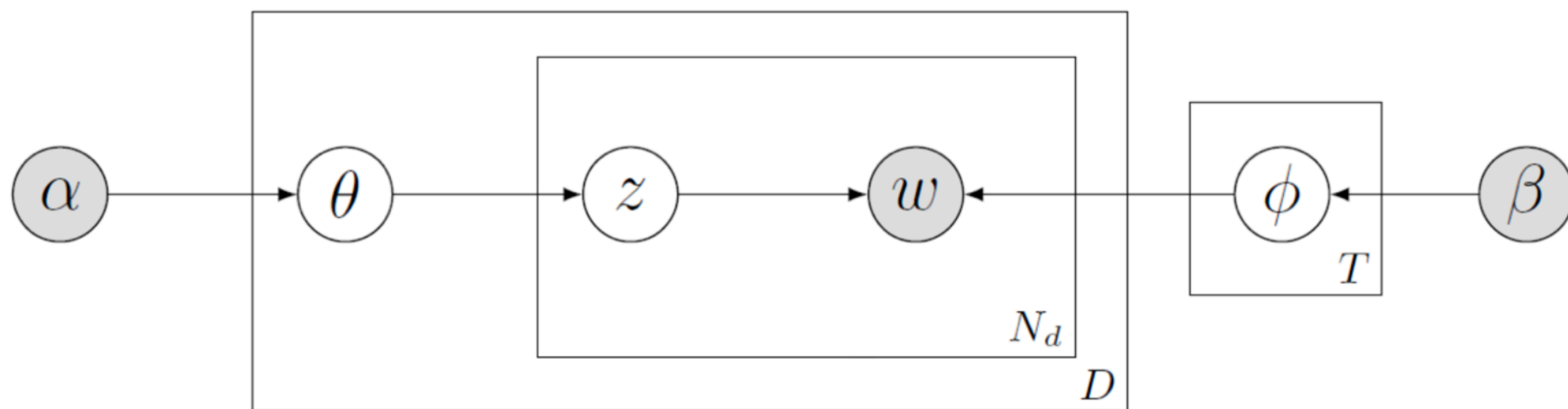
Graphical Model



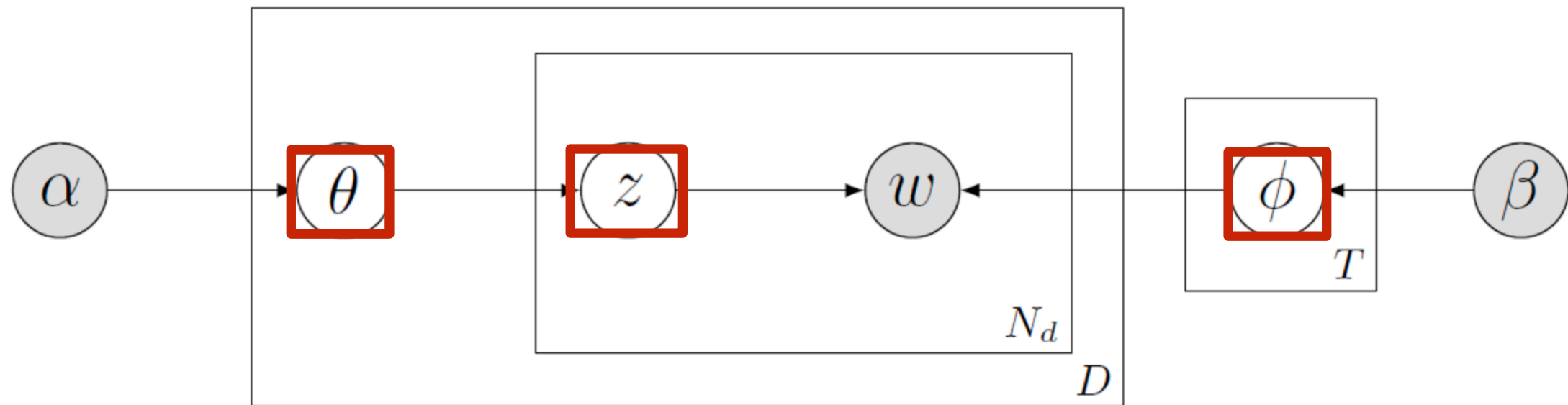
- Nodes are random variables
- Gray nodes are observed variables or constant
- Edge = conditional dependency
- Plate notation : variables are repeated

LDA generative story

1. For each topic $j \in \{1, \dots, T\}$ draw a sample of words ϕ_j from a $\text{Dirichlet}_W(\beta)$
2. For each document $d \in D$
 - ▶ Draw a topic distribution θ_d from $\text{Dirichlet}_T(\alpha)$
 - ▶ For each word at position n in document d
 - Draw a topic $z_{d,n}$ from $\text{Multinomial}_T(\theta_d)$
 - Draw a word $w_{d,n}$ from $\text{Multinomial}_W(\phi_{z_{d,n}})$



Estimating the model



- ▶ Approximate posterior inference algorithms
 - ▶ Mean field variational methods
 - ▶ Expectation propagation
 - ▶ **Collapsed Gibbs sampling***
 - ▶ Collapsed variational inference
 - ▶ Online variational inference

Take away message

- ▶ The estimation of the parameters in the model is performed based on the observed data and the parameters
- ▶ Clustering can be performed using LDA by choosing the most representative topic and fixing the desired number of topics as the number of clusters

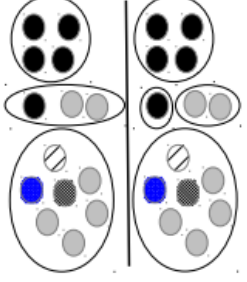
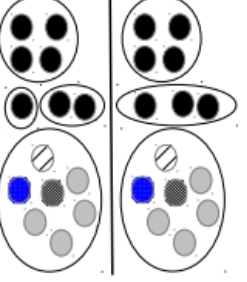
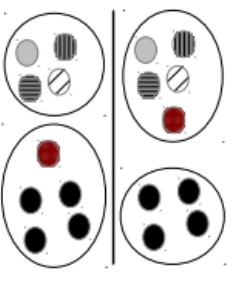
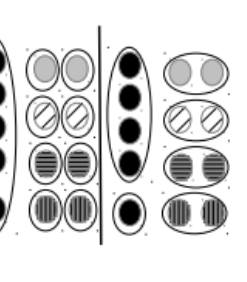
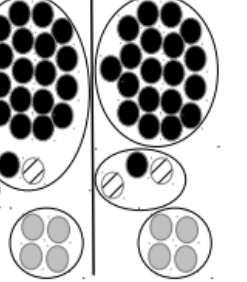
More about clustering

- ▶ Many algorithms are available for clustering, however many of them are not suitable for text
- ▶ Classical algorithms usually perform well, but state-of-the-art algorithms can perform much better if they are well parametrized (not free lunch)
- ▶ Clustering is a hard task and few algorithms scale well to huge text collections. If hierarchies are needed, bottom-up or top-down strategies can be applied (combined with the presented algorithms)
- ▶ There exist problems in which labels are needed!!! LDA is a good solution however other simpler algorithms can do the job (STC, Lingo, etc)
- ▶ As any other data mining task, the best clustering algorithm will be drawn by the problem and not by choosing the most popular one

How to evaluate clustering?

- ▶ Evaluation is a hard task.
- ▶ Best scenario: an annotated collection is available
 - ▶ Pairs of documents are used to evaluate the partitions by asking if they belong to the same partition in the annotated data and in the obtained partition (for any clustering algorithm)
- ▶ Extreme cases are hard to evaluate
 - ▶ All documents belong to just one cluster
 - ▶ All documents belong to individual clusters

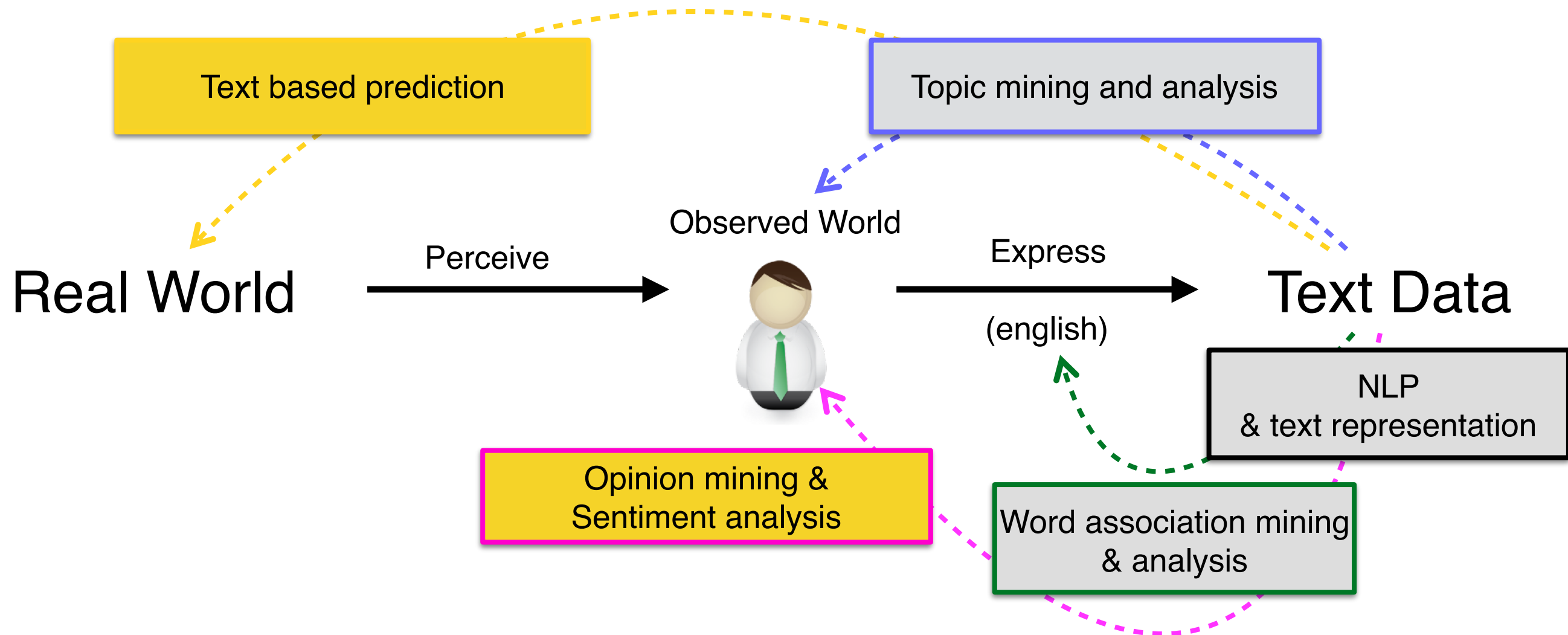
Comparison of clustering evaluation metrics

	C. Homogeneity			C. Completeness			Rag Bag			C. size vs q.			Unbalanced			4 + 1 F.C.
																
Purity	0.71	0.79	✓	0.79	0.79	✗	0.56	0.56	✗	1.00	1.00	✗	0.96	0.96	✗	✗
Inv. Purity	0.79	0.79	✗	0.79	0.79	✗	1.00	1.00	✗	0.69	0.92	✓	0.96	0.96	✗	✗
F&M	0.47	0.49	✓	0.47	0.53	✓	0.61	0.61	✗	0.85	0.85	✗	0.95	0.94	✗	✗
RandIndex	0.68	0.70	✓	0.68	0.70	✓	0.72	0.72	✗	0.95	0.95	✗	0.94	0.94	✗	✗
Adj.RandIndex	0.25	0.28	✓	0.24	0.31	✓	0.40	0.40	✗	0.80	0.80	✗	0.79	0.79	✗	✗
Jaccard	0.31	0.33	✓	0.31	0.36	✓	0.38	0.38	✗	0.71	0.71	✗	0.90	0.89	✗	✗
F-measure	0.71	0.79	✓	0.79	0.79	✗	0.56	0.56	✗	1.00	1.00	✗	0.96	0.96	✗	✗
P_{b^3}	0.60	0.69	✓	0.69	0.69	✗	0.49	0.56	✓	1.00	1.00	✗	0.93	0.95	✓	✗
R_{b^3}	0.70	0.70	✗	0.71	0.76	✓	1.00	1.00	✗	0.69	0.88	✓	0.96	0.93	✗	✗
F_{b^3}	0.64	0.69	✓	0.70	0.72	✓	0.55	0.71	✓	0.82	0.93	✓	0.94	0.93	✗	✗
$P_{b^3}^{mod}$	0.60	0.69	✓	0.69	0.69	✗	0.49	0.56	✓	1.00	1.00	✗	0.93	0.95	✓	✗
$R_{b^3}^{mod} (\vec{x} = 3)$	0.45	0.45	✗	0.56	0.57	✓	1.00	1.00	✗	0.46	0.77	✓	0.93	0.86	✗	✗
$F_{b^3}^{mod\&0.9}$	0.58	0.66	✓	0.67	0.68	✓	0.52	0.58	✓	0.90	0.97	✓	0.93	0.95	✓	✓
$F_{b^3}^{0.9}$	0.61	0.70	✓	0.69	0.70	✓	0.52	0.58	✓	0.96	0.99	✓	0.93	0.94	✓	✓

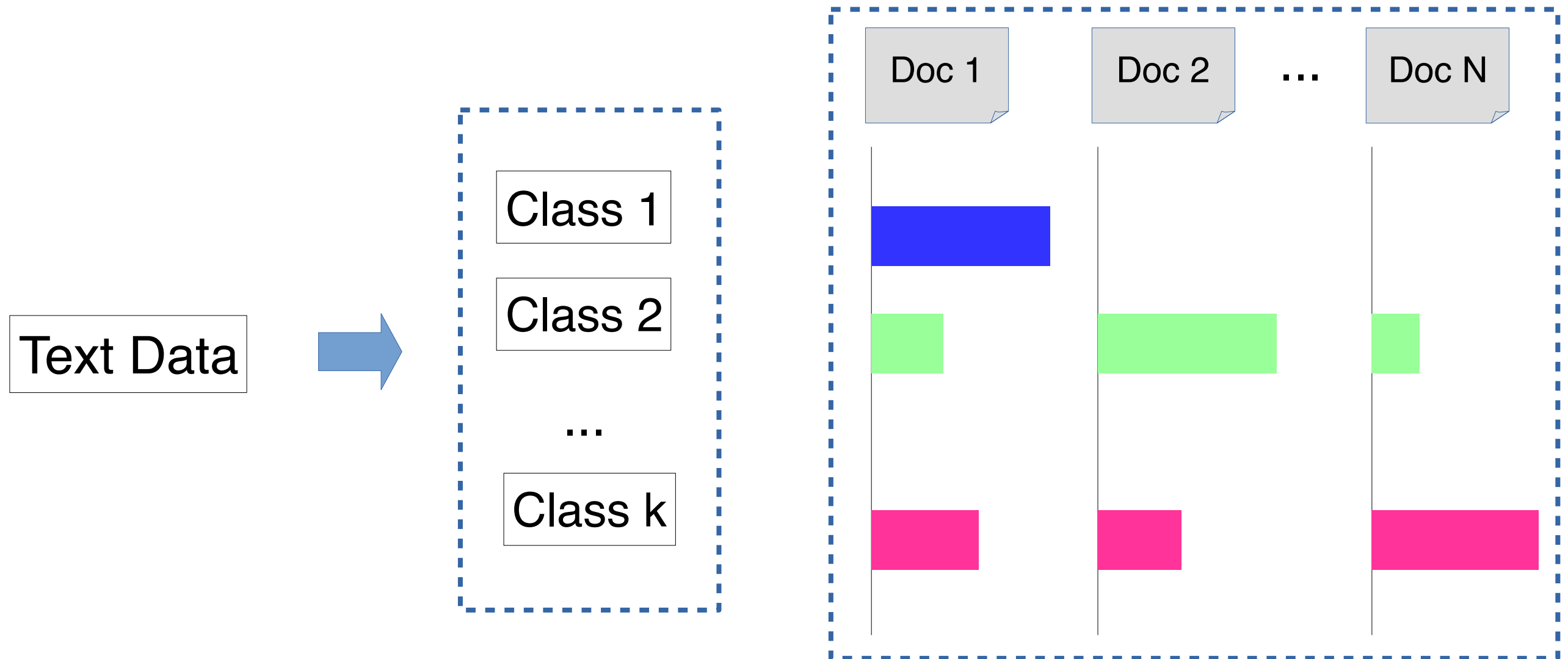
Take away message

- ▶ Evaluate if you have annotated data
 - ▶ Metric selection is an important issue
 - ▶ Try to understand the addressed problem to select a metric adapted to it
- ▶ Tools also implement evaluation metrics
 - ▶ Clustering performance evaluation module in scikit-learn
 - ▶ Clusteval in R

Landscape of Text Mining and Analytics



Task of Topic Mining and Analysis



Supervised problem
(classification)

Topic mining and Analysis

- ▶ Group of documents (unsupervised)
 - ▶ Clustering
- ▶ **Group of documents but into classes (supervised)**
 - ▶ **Classification**
- ▶ Group of words - dimensionality reduction (We are going to see them in the unsupervised part)
 - ▶ NMF, LDA, PLSA, etc.

Text classification or categorization

- ▶ Given the following
 - ▶ A set of predefined categories, possibly forming a hierarchy and often
 - ▶ A training set of labeled text objects
- ▶ Task: Classify a text object into one or more of the categories

Examples of text categorization

- ▶ Text objects can vary (e.g., documents, passages, or collections of text)
- ▶ Categories can also vary
 - ▶ Internal categories that characterize a text object (e.g., topical categories, sentiment categories)
 - ▶ External categories that characterize an entity associated with the object (e.g., author attribution or any other meaningful categories associated with text data)
- ▶ Some examples of applications
 - ▶ News categorization, literature article categorization (e.g., MeSH annotations)
 - ▶ Spam email detection/filtering
 - ▶ Sentiment categorization of product reviews or tweets
 - ▶ Automatic email sorting/routing
 - ▶ Author attribution

Variants of problem formulation

- ▶ Binary categorization: only two categories
 - ▶ Retrieval (relevant or not)
 - ▶ Spam filtering (spam or not)
 - ▶ Opinion (negative or positive)
- ▶ K-category categorization: more than two categories
 - ▶ Topic categorization (sports, science, travel, business, etc.)
 - ▶ Email routing (folder 1, folder 2, etc.)
- ▶ Hierarchical categorization: categories for a hierarchy
- ▶ Joint categorization: multiple related categorization tasks done in a joint manner

Why text categorization?

- ▶ To enrich text representation (more understanding of text)
 - ▶ Text can now be represented in multiple levels (keywords+categories)
 - ▶ Semantic categories assigned can be directly or indirectly useful for an application
 - ▶ Semantic categories facilitate aggregation of text content (e.g., aggregating all positive/negative opinions about a product)
- ▶ To infer properties of entities associated with text data (discovery of knowledge about the world)
 - ▶ As long as an entity can be associated with text data, we can always use the text data to help to categorize the associated entities
 - ▶ E.g., discovery of non-native speakers of a language; prediction of party affiliation based on a political speech

Categorization methods: manual

- ▶ Determine the category based on rules that are carefully designed to reflect the domain knowledge about the categorization problem
- ▶ Works well when
 - ▶ The categories are very well defined
 - ▶ Categories are easily distinguished based on surface features in text (e.g., special vocabulary is known to only occur in a particular category)
 - ▶ Sufficient domain knowledge is available to suggest many effective rules
- ▶ Problems
 - ▶ Labor intensive -> doesn't scale up well
 - ▶ Can't handle uncertainty in rules; rules may be inconsistent -> not robust
- ▶ Both problems can be solved/alleviated by using machine learning

Categorization methods: automatic

- ▶ Use human experts to
 - ▶ Annotate data sets with category labels -> training data
 - ▶ Provide a set of features to represent each text object that can potentially provide a "clue" about the category
 - ▶ Use machine learning to learn "soft rules" for separating different categories
 - ▶ Figure out which features are most useful for separating different categories
 - ▶ Optimally combine the features to minimize the errors of categorization on the training data
 - ▶ The trained classifier can then be applied to a new text object to predict the most likely category (that a human expert would assign to it)

Machine learning for text categorization

- ▶ General setup: learn a classifier $f: X \rightarrow Y$
 - ▶ Input: X = all objects; Output: Y all categories
 - ▶ Learn a classifier function, $f: X \rightarrow Y$, such that $f(x)=y$ gives correct category for x (correct is based on the training data)
- ▶ All methods
 - ▶ Rely on discriminative features of text objects to distinguish categories
 - ▶ Combine multiple features in a weighted manner
 - ▶ Adjust weights on features to minimize errors on the training data
- ▶ Different methods tend to vary in
 - ▶ Their way of measuring the errors on the training data (may optimize a different objective/loss/cost function)
 - ▶ Their way to combining features (e.g. linear vs non-linear)

Generative vs Discriminative Classifiers

- ▶ Generative classifiers (learn what the data "looks" like in each category)
 - ▶ Attempt to model $p(X,Y)=p(Y)P(X|Y)$ and compute $p(Y|X)$ based on $p(X|Y)$ and $p(Y)$ by using the Bayes rule
 - ▶ Objective function is likelihood, thus indirectly measuring training error
 - ▶ E.g., Naive Bayes
- ▶ Discriminative classifiers (learn what features separate categories)
 - ▶ Attempt to model $p(Y|X)$ directly
 - ▶ Objective function directly measures errors of categorization on training data
 - ▶ E.g., logistic regression, Support vector machines (SVM), k-Nearest Neighbors (kNN)

Text Categorization with Naïve Bayes

- ▶ Consider each category independently as a class c (for the multiple class setting)
 - ▶ Example d – document
 - ▶ Feature w – word or term

$$score(c) = \log \frac{P(c|d)}{P(\sim c|d)} = \sum_{w \in d} \log \frac{P(w|c)}{P(w|\sim c)} + \log \frac{P(C)}{P(\sim C)}$$

- ▶ Classify c if $score(c) > \theta$
 - Typically a specifically tuned threshold for each class, due to inaccuracy of the probabilistic estimate of $P(d|c)$ with given training statistics and independence assumption,
 - .. but a biased probability estimate for c may still correlate well with the classification decision

Nearest-Neighbor Learning Algorithm

- ▶ Learning is just storing the representations of the training examples in data set D
- ▶ Testing instance x :
 - ▶ Compute similarity between x and all examples in D
 - ▶ Assign x the category of the most similar examples in D
- ▶ Does not explicitly compute a generalization or category prototypes (i.e., no “modeling”)
- ▶ Also called:
 - ▶ Case-based
 - ▶ Memory-based
 - ▶ Lazy learning

K Nearest Neighbor for Text

- ▶ Training:
 - ▶ For each each training example $\langle x, c(x) \rangle \in D$
 - Compute the corresponding TF-IDF vector, d_x , for document x
- ▶ Test instance y :
 - ▶ Compute TF-IDF vector d for document y
 - ▶ For each $\langle x, c(x) \rangle \in D$
 - Let $s_x = \cos(d, d_x)$
 - ▶ Sort examples, x , in D by decreasing value of s_x
 - ▶ Let N be the first k examples in D . (get most similar neighbors)
 - ▶ Return the majority class of examples in N

Simple but powerful in very large collections!

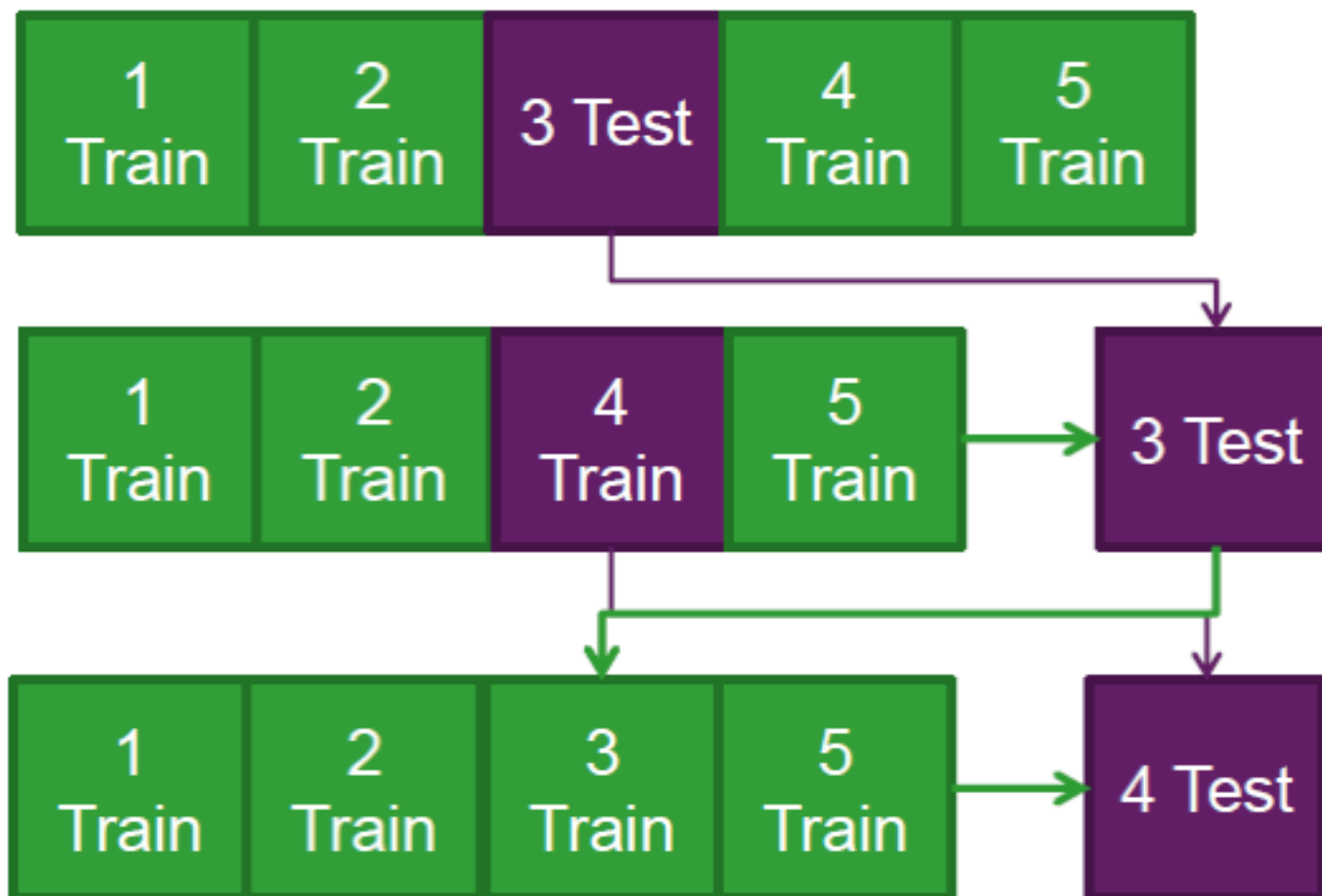
KNN discussion

- ▶ No feature selection necessary
- ▶ No training necessary
- ▶ Scales well with large number of classes/documents
- ▶ Don't need to train n classifiers for n classes
- ▶ Classes can influence each other
- ▶ Small changes to one class can have ripple effect
- ▶ Done naively, very expensive at test time

Evaluation

- ▶ Recall: Fraction of docs in class i classified correctly
- ▶ Precision: Fraction of docs assigned class i that are actually about class i
- ▶ Accuracy: $(1 - \text{error rate})$ Fraction of docs classified correctly
- ▶ Other metrics, but again the problem define the appropriate metrics

Cross validation example



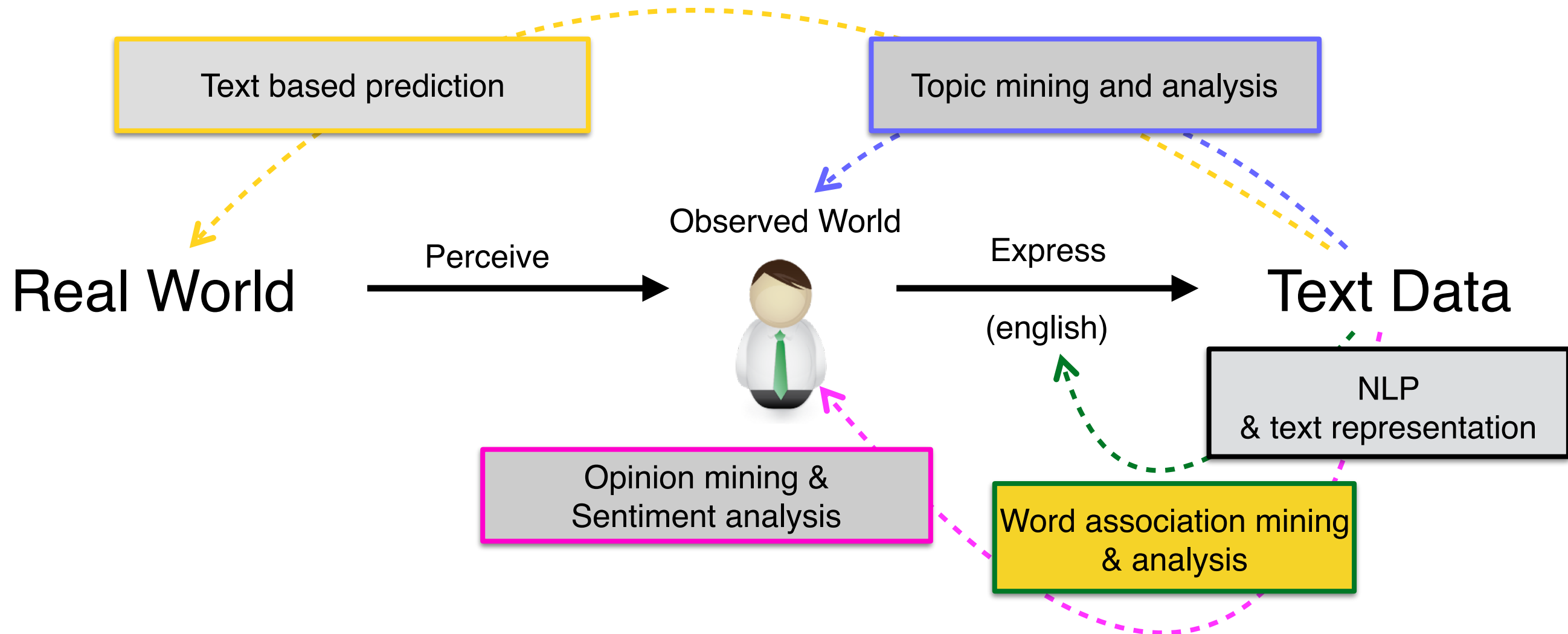
- ▶ Split the data into 5 samples
- ▶ Fit a model to the training samples and use the test sample to calculate a CV metric.
- ▶ Repeat the process for the next sample, until all samples have been used to either train or test the model

Examples within the landscape of Text Mining and Analytics

http://scikit-learn.org/stable/auto_examples/text/document_clustering.html

[http://scikit-learn.org/stable/auto_examples/text/
document_classification_20newsgroups.html](http://scikit-learn.org/stable/auto_examples/text/document_classification_20newsgroups.html)

Landscape of Text Mining and Analytics



```
import nltk.collocations
import collections
```

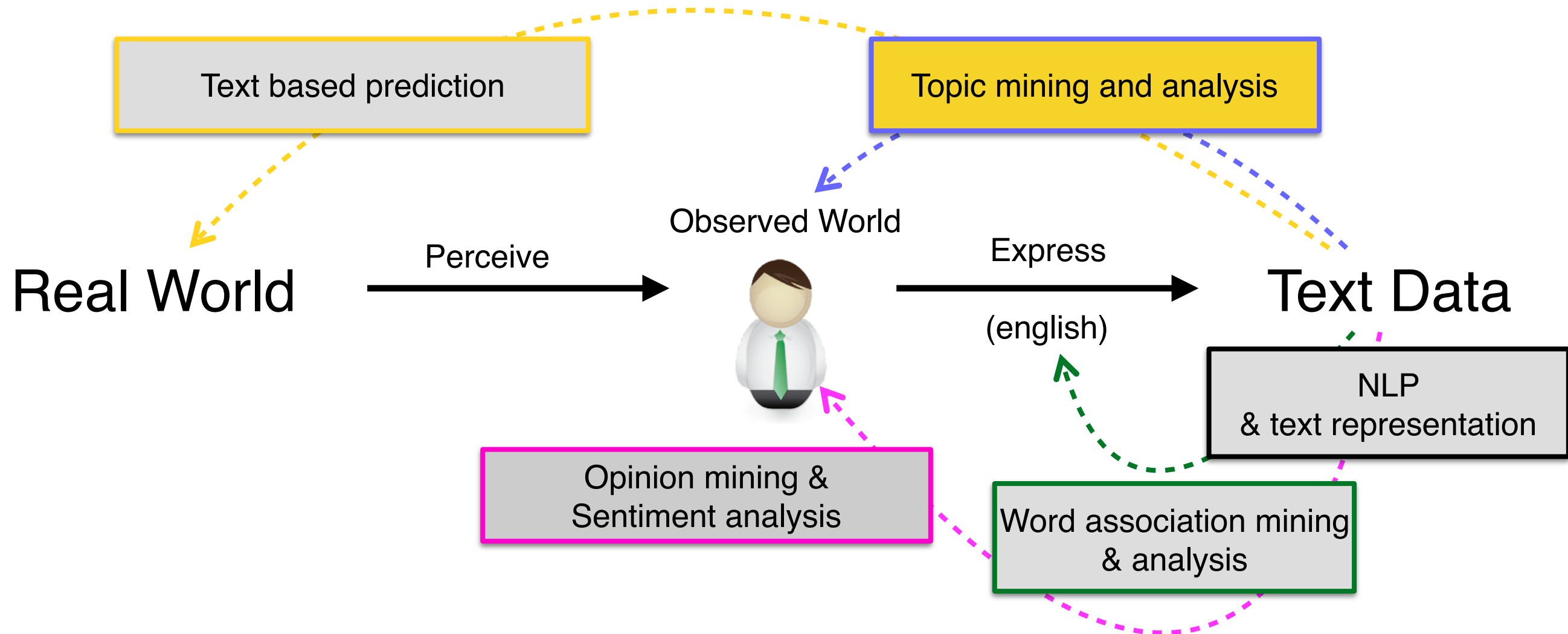
```
f = open("full.txt")
documents = " ".join([line for line in f.readlines()])
f.close()
```

```
bgm = nltk.collocations.BigramAssocMeasures()
finder = nltk.collocations.BigramCollocationFinder.from_words(documents.split())
ignored_words = nltk.corpus.stopwords.words('english')
finder.apply_word_filter(lambda w: len(w) < 3 or w.lower() in ignored_words)
```

```
finder.nbest(bgm.raw_freq, 10)
finder.nbest(bgm.pmi, 10)
finder.nbest(bgm.likelihood_ratio, 10)
finder.nbest(bgm.chi_sq, 10)
finder.nbest(bgm.dice, 10)
finder.nbest(bgm.fisher, 10)
finder.nbest(bgm.jaccard, 10)
finder.nbest(bgm.mi_like, 10)
finder.nbest(bgm.poisson_stirling, 10)
finder.nbest(bgm.student_t, 10)
```

```
scored = finder.score_ngrams(bgm.pmi)
```

Landscape of Text Mining and Analytics




```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.decomposition import NMF, LatentDirichletAllocation
from sklearn.cluster import KMeans
```

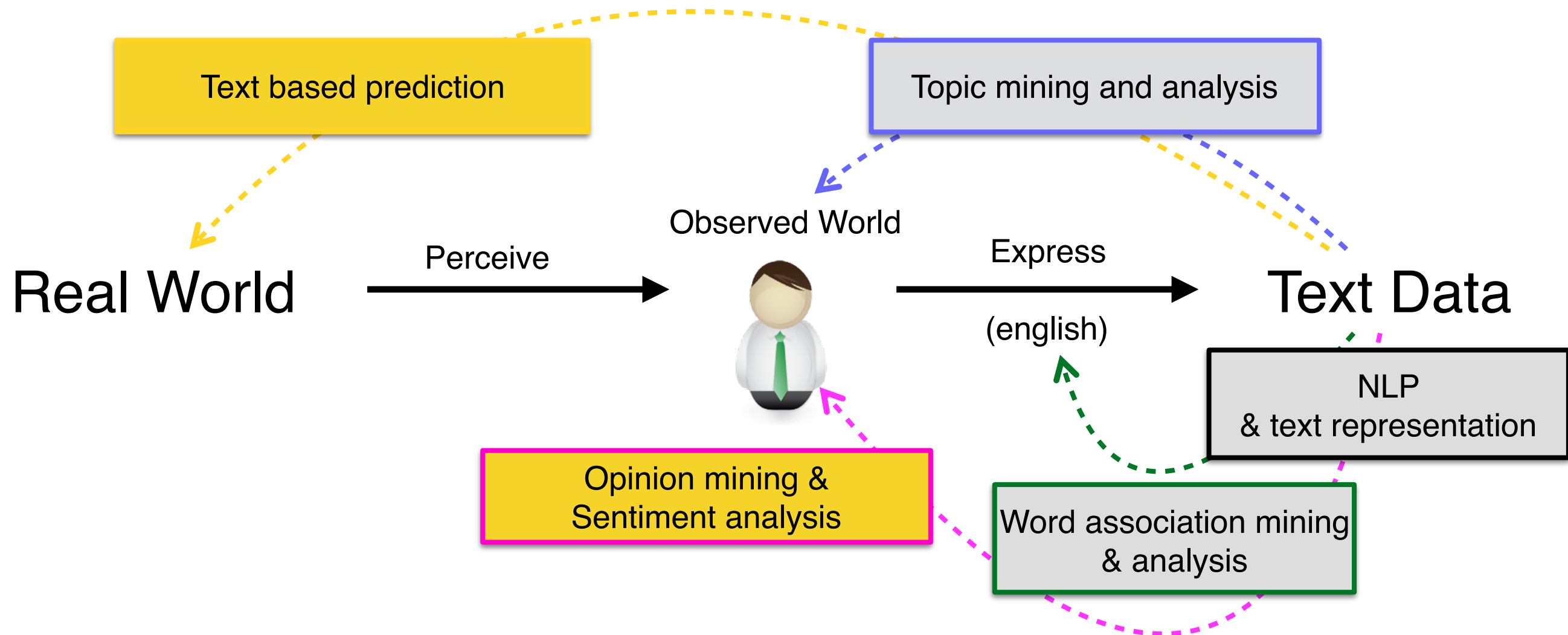
```
f = open("full.txt.head")
documents = [line for line in f.readlines()]
f.close()
```

```
tf_v = CountVectorizer(max_df=0.95, min_df=2, max_features=100000, stop_words='english')
tfidf_v = TfidfVectorizer(max_df=0.95, min_df=2, max_features=100000, stop_words='english')

tf = tf_v.fit_transform(documents)
tfidf = tfidf_v.fit_transform(documents)
```

```
nmf = NMF(n_components=10, random_state=1, alpha=.1, l1_ratio=.5).fit(tfidf)
km = KMeans(n_clusters=10, init='k-means++', max_iter=100, n_init=1).fit(tfidf)
lda = LatentDirichletAllocation(n_topics=10, max_iter=5, learning_method='online',
learning_offset=50., random_state=0).fit(tf)
```

Landscape of Text Mining and Analytics



```
from sklearn.svm import SVC
from sklearn.feature_extraction.text import TfidfVectorizer
import random
```

```
from nltk.corpus import movie_reviews
docs = [(list(movie_reviews.words(fileid)), category)
         for category in movie_reviews.categories()
         for fileid in movie_reviews.fileids(category)]
random.shuffle(docs)
X,y= [" ".join(w[0]) for w in docs],[1 if w[1] == 'pos' else 0 for w in docs]
```

```
tfidf_v = TfidfVectorizer(max_df=0.95, min_df=2, max_features=100000, stop_words='english')

tfidf = tfidf_v.fit_transform(X)
```

```
clf = SVC()
clf.fit(tfidf, y)
```

```
tfidf_test = tfidf_v.transform("One of my all time favorites. Shawshank Redemption is a very moving story about hope and the power of friendship. The cast is first rate with everyone giving a great performance. Tim Robbins and Morgan Freeman carry the movie, but Bob Gunton and Clancy Brown are perfect as the Warden Norton and prison guard captain Hadley respectively. And James Whitmore's portrayal of an elderly inmate Brooks is moving. The screenplay gives almost every actor at least one or more memorable lines through out the film. As well as a very surprising twist near the end that almost knocked me out of my chair. If you have not seen this movie rent it or better yet buy it. As I bet you'll want to see this one more than once.".split())
```

```
print(clf.predict(tfidf_test))
```