IR Meets Data Mining, Machine Learning, and Natural Language Processing

Alejandro Moreo and Fabrizio Sebastiani

Artificial Intelligence for Media and Humanities Lab
Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
56124 Pisa, Italy
E-mail: {fabrizio.sebastiani}@isti.cnr.it

13th European Summer School in Information Retrieval (ESSIR 2022)
Lisbon, PT – July 18-22, 2022

Introduction

- Today we won't pronounce the words "search" and "query" ...
- We'll deal with tasks that do not belong to "core IR", but that lie at the intersection between IR and DM / ML / NLP
- These tasks have many commonalities with search
  - They allow users to access non-structured information (typically: text)
  - They are tedious / expensive / time-consuming / difficult to carry out manually
  - Automatic solutions for them can be devised via a combination of machine learning and automatic content analysis

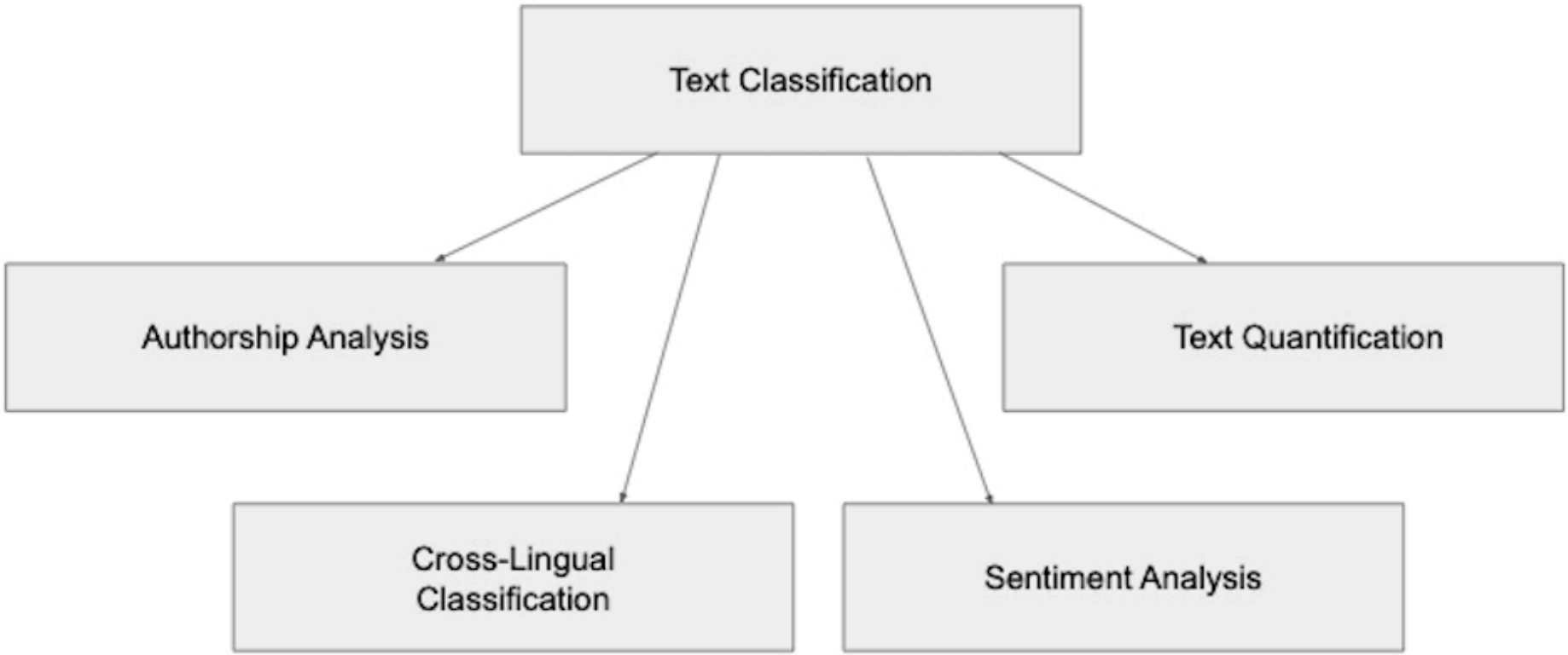
Plan of today's lecture

1. Text classification
   - Basic TC via "classic" machine learning [30 min - Fabrizio]
   - Advanced TC via "deep learning" [30 min - Alejandro]
2. Authorship analysis [30 min - Fabrizio]

Break

3. Cross-lingual text classification [30 min - Alejandro]
4. Sentiment analysis [30 min - Fabrizio]
5. Text quantification [30 min - Alejandro]

Plan of today's lecture



Part I :
Text Classification

Introduction to TC

- Many text analysis tasks can be framed as classification tasks, i.e., as the task of predicting / hypothesizing / deciding to which among a predefined finite set of groups ("classes", or "categories") a data item belongs to
- Classification is formulated as the task of generating a hypothesis (or "classifier", or "model")

$$h : \mathcal{D} \rightarrow \mathcal{C}$$

where $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, ...\}$ is a domain of data items and $\mathcal{C} = \{c_1, ..., c_n\}$ is a finite set of classes (the classification scheme, or codeframe)

Applications of TC

- Assigning newspaper articles to the page (**HomeNews**, **Sports**, **Lifestyles**, etc.) where they should appear
- Assigning subject codes (from a predefined taxonomy) to scientific papers
  $\rightarrow$ codeframe is the taxonomy of subject codes
- Partitioning emails into **Legitimate** and **Spam** (Spam Filtering)
- Coding textual answers returned to open questions in questionnaires (e.g., in opinion research, market research, customer relationship management)
  $\rightarrow$ codeframe is the set of codes of interest (Survey Coding)
- Marking a textual comment (on a product, on a political candidate, etc.) as conveying a positive or a negative opinion about its subject
  $\rightarrow$ codeframe is {**Positive**,**Negative**,**Neutral**} (Sentiment Classification)
- Detecting whether a suspect (e.g., John) is the author of an anonymous text
  $\rightarrow$ codeframe is {**John**,**NotJohn**} (Authorship Verification)
- ...

Introduction to TC

- Can classification be automated?
- Can we build tools that support the work of humans who need to classify text?
- Problems (as in IR):
  - unlike other types of data (e.g., factual data, numerical measurements, etc.), text requires (subjective) interpretation
    $\rightarrow$ all the above tasks are non-deterministic
  - the variety of linguistic devices that humans use in order to convey meaning is bewildering
  - language use differs across people
  - language keeps evolving
- Programming "if-then" rules that automatically classify text is thus
  - difficult
  - bound to lead to software characterized by low accuracy
  - not economical (in terms of both creation costs and maintenance costs)
  - too slow for fast-emerging needs

Introduction to TC

- Idea: set up a software that *learns* to carry out the task from examples in which the task has been performed correctly by competent humans
- Example:
  - Task: Assigning subject codes (from a predefined taxonomy) to scientific papers
  - Idea: The software learns to do this by looking at a set of papers whose codes have been assigned by experts
  - Consequence: This software must try to understand the characteristics that make a paper suitable for assigning it a certain code
- This is the core idea behind supervised machine learning
- This course is about
  - formulating text analysis tasks in terms of text classification
  - using machine learning technology to design, implement, and test the resulting text classification systems

Text Classification

What Classification is and is not

- Classification (a.k.a. "categorization"): a ubiquitous enabling technology in data science (or: the "mother" of all machine learnable tasks); studied within pattern recognition, statistics, and machine learning
- Different from clustering, where the groups ("clusters") and their number are not known in advance
- The membership of a data item into a class must not be determinable with certainty (e.g., predicting whether a natural number belongs to **Odd** or **Even** is not classification); classification always involves a subjective judgment
- In text classification, data items are textual (e.g., news articles, treatises, emails, tweets, product reviews, sentences, questions, queries, etc.) or partly textual (e.g., Web pages)

Main Types of Classification

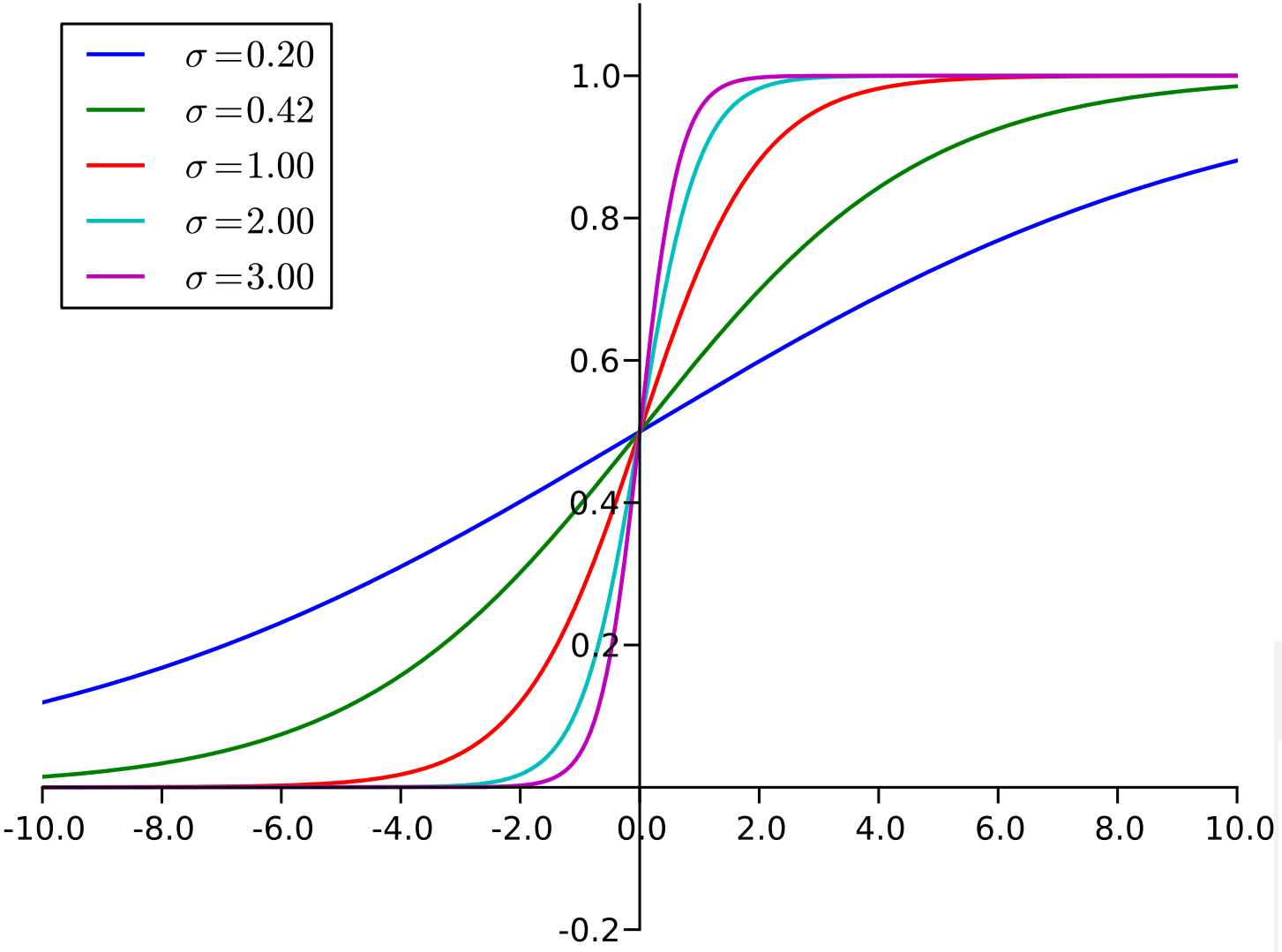
- Binary classification: $h : \mathcal{D} \rightarrow \mathcal{C}$ (each item belongs to exactly one class) and $\mathcal{C} = \{c_1, c_2\}$
  - E.g., assigning emails to one of {**Spam**, **Legitimate**}
- Single-Label Multi-Class (SLMC) classification: $h : \mathcal{D} \rightarrow \mathcal{C}$ (each item belongs to exactly one class) and $\mathcal{C} = \{c_1, ..., c_n\}$, with $n > 2$
  - E.g., assigning news articles to one of {**HomeNews**, **International**, **Entertainment**, **Lifestyles**, **Sports**}
- Multi-Label Multi-Class (MLMC) classification: $h : \mathcal{D} \rightarrow 2^{\mathcal{C}}$ (each item may belong to zero, one, or several classes) and $\mathcal{C} = \{c_1, ..., c_n\}$, with $n > 1$
  - E.g., assigning computer science articles to classes in the ACM Classification System
  - May be solved as n independent binary classification problems
- Ordinal classification (OC): as in SLMC, but for the fact that there is a total order $c_1 \preceq ... \preceq c_n$ on $\mathcal{C} = \{c_1, ..., c_n\}$
  - E.g., assigning product reviews to one of {**Disastrous**, **Poor**, **SoAndSo**, **Good**, **Excellent**}

Hard Classification and Soft Classification

- The definitions above denote "hard classification" (HC)
- "Soft classification" (SC) denotes the task of predicting a score for each pair $(\mathbf{x}, c)$, where the score denotes the { probability / strength of evidence / confidence } that $\mathbf{x}$ belongs to c
  - E.g., a probabilistic classifier outputs "posterior probabilities" $\Pr(c|\mathbf{x}) \in [0, 1]$
  - E.g., the AdaBoost classifier outputs scores $s(\mathbf{x}, c) \in (-\infty, +\infty)$ that represent its confidence that $\mathbf{x}$ belongs to c
  - When scores are not probabilities, they can be converted into probabilities via the use of a sigmoidal function; e.g., the logistic function:

$$\Pr(c|\mathbf{x}) = \frac{1}{1 + e^{\sigma h(\mathbf{x}, c) + \beta}}$$

Hard Classification and Soft Classification (cont'd)



Hard Classification and Soft Classification (cont'd)

- In the binary case, hard classification often consists of
  1. Training a soft classifier that returns scores $s(\mathbf{x}, c)$
  2. Picking (via parameter optimization techniques) a threshold t such that
     - $s(\mathbf{x}, c) \geq t$ is interpreted as predicting c_1
     - $s(\mathbf{x}, c) < t$ is interpreted as predicting c_2
- In soft classification, scores are used for ranking; e.g.,
  - ranking items for a given class
  - ranking classes for a given item
- HC is used for fully autonomous classifiers, while SC is used for interactive classifiers (i.e., with humans in the loop)

Dimensions of Classification

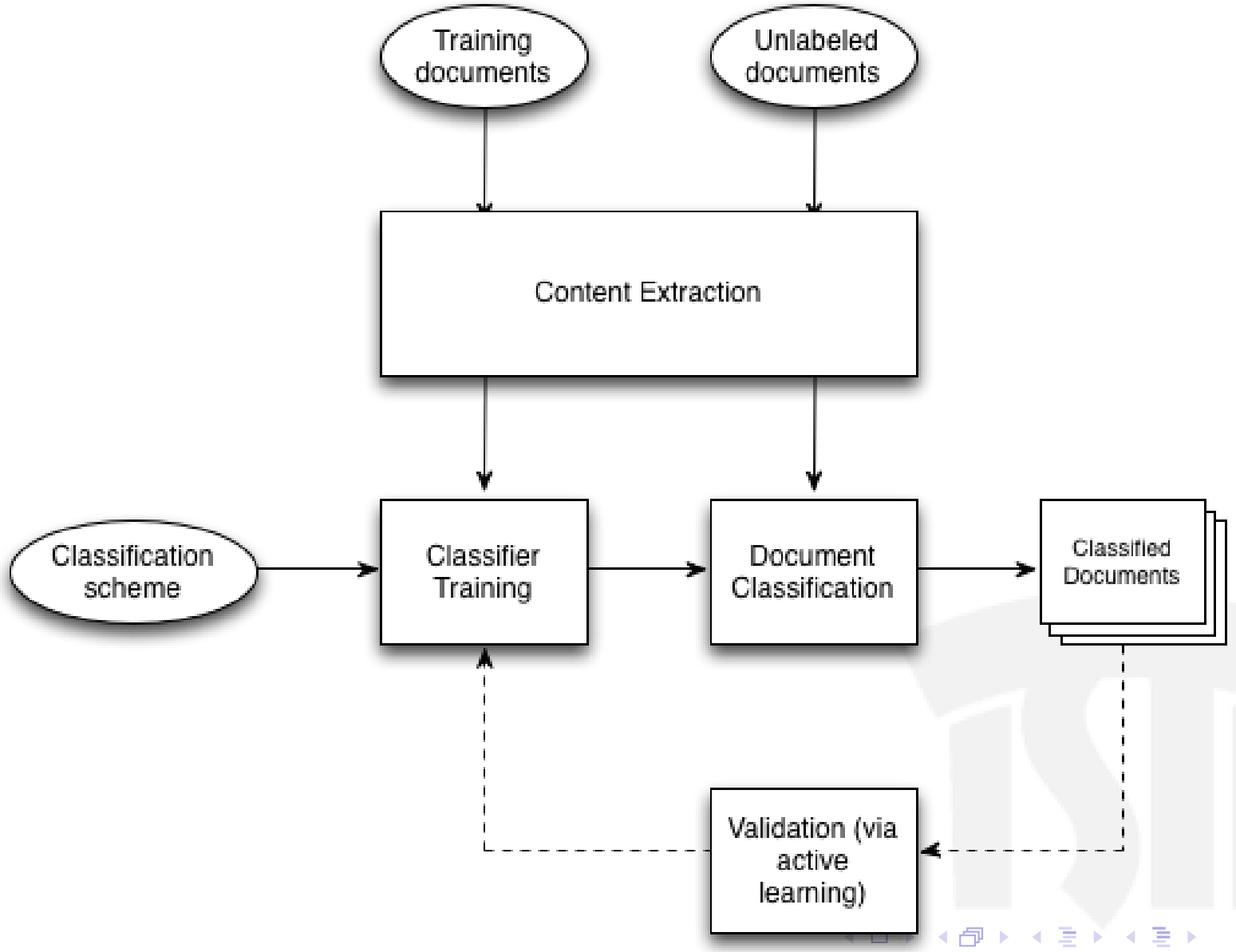
- Text classification may be performed according to several dimensions ("axes") independent of each other
- by topic ; by far the most frequent case, its applications are ubiquitous
- by sentiment ; useful in market research, online reputation management, customer relationship management, social sciences, political science
- by author (a.k.a. "authorship attribution"), or by native language ("native language identification"); useful in philology, forensics, and cybersecurity
- ...

Text Classification

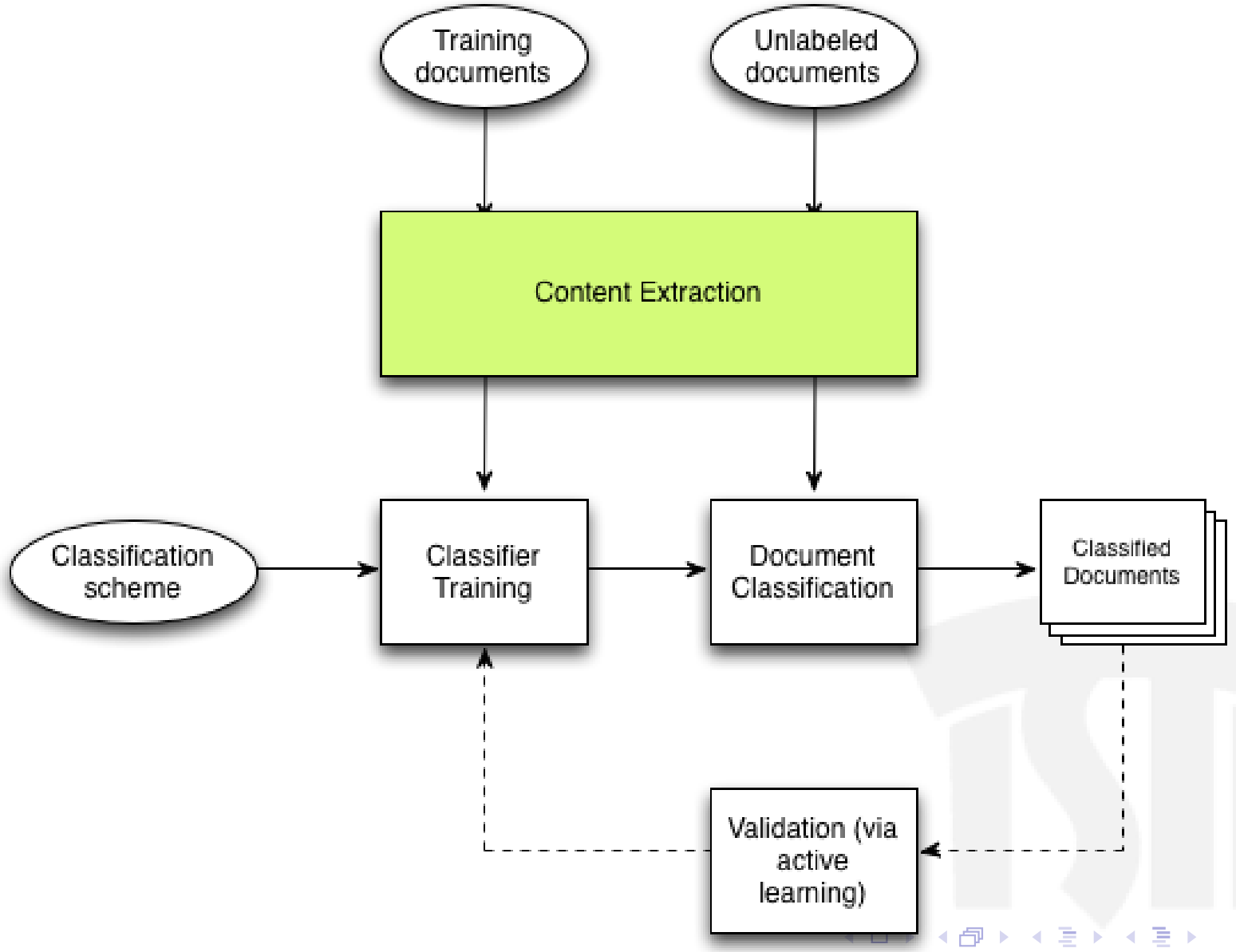
The Supervised Learning Approach to Classification

- Supervised learning (SL) approach
  - A generic (task-independent) learning algorithm is used to train a classifier from a set of manually classified examples
  - The classifier learns, from these training examples, the characteristics a new text should have in order to be assigned to class c
- Advantages:
  - Annotating / locating training examples cheaper than writing classification rules
  - Easy update to changing conditions (e.g., addition of new classes, deletion of existing classes, shifted meaning of existing classes, etc.)

The Supervised Learning Approach to Classification



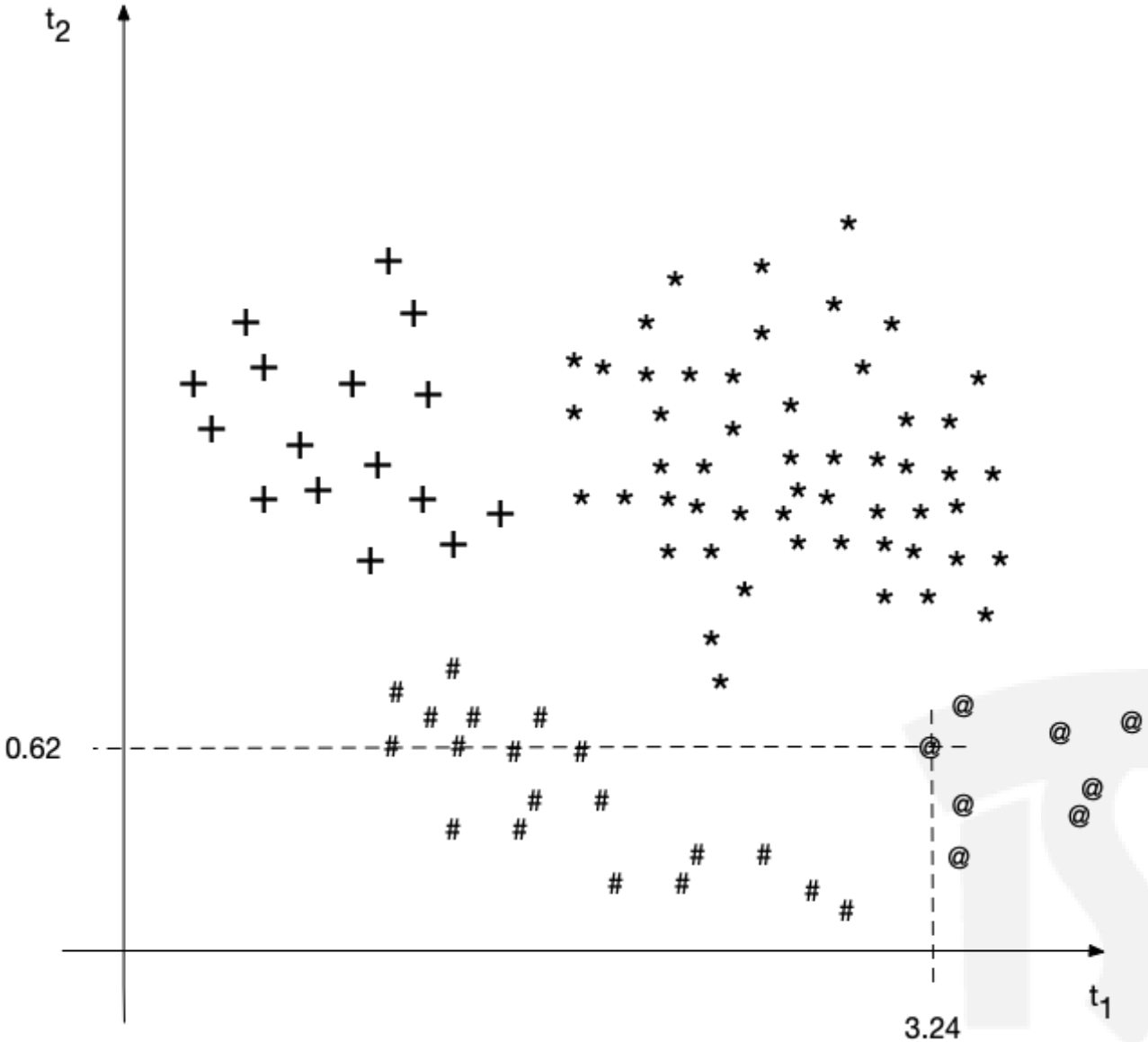
The Supervised Learning Approach to Classification



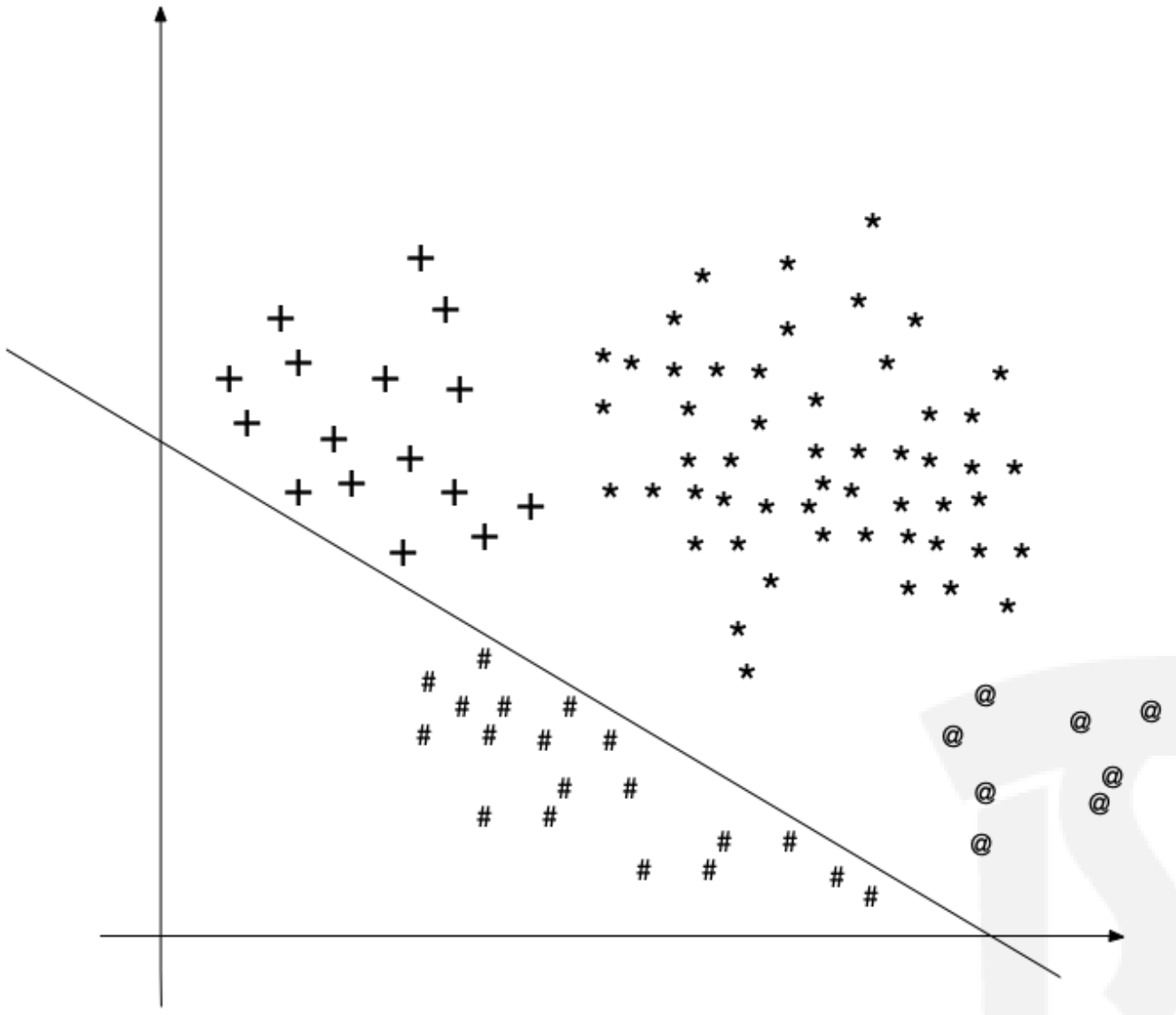
Representing Text for Classification Purposes

- In order to be input to a learning algorithm (or a classifier), all training (or unlabeled) documents are converted into vectors in a common vector space
- The dimensions of the vector space are called features (or terms, or covariates), and the number K of features used is called the dimensionality of the vector space
- In order to generate a vector-based representation for a set of documents $D = L \cup U$ (with L the labelled training set and U the unlabelled set), the following steps need to be taken
  1. Feature Design and Extraction
  2. (Feature Selection *or* Feature Synthesis)
  3. Feature Weighting

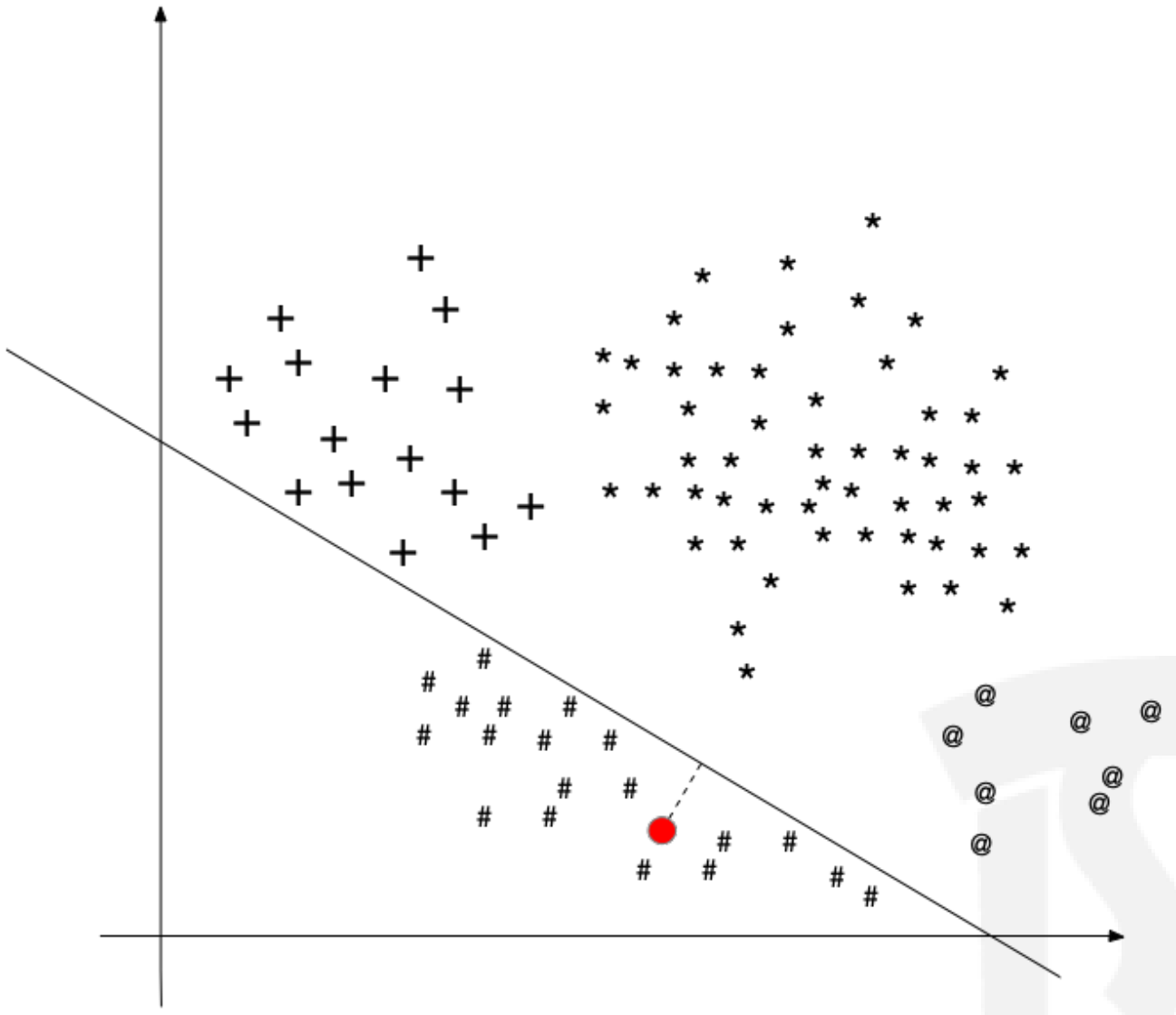
Representing Text for Classification Purposes



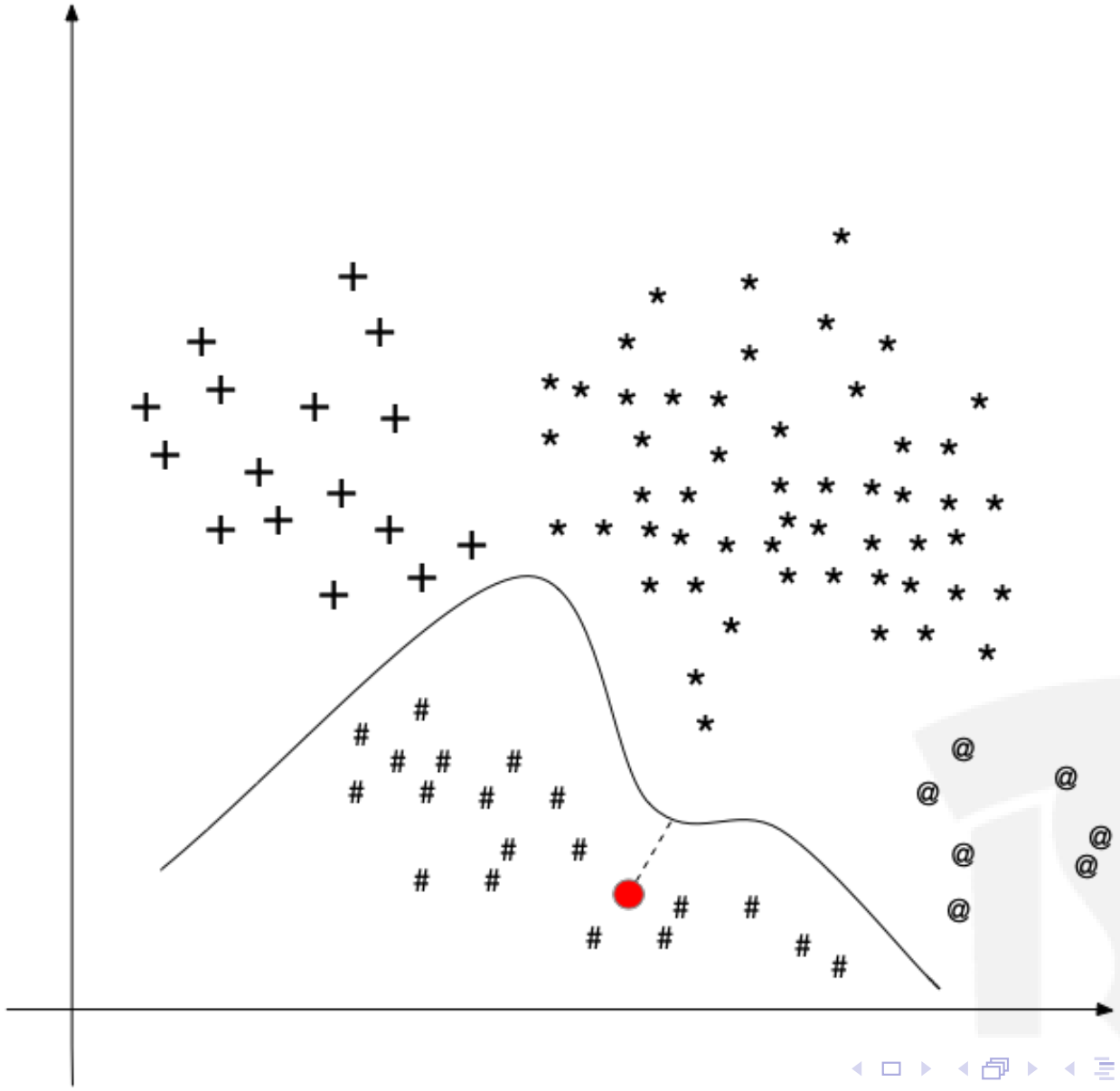
Representing Text for Classification Purposes



Representing Text for Classification Purposes



Representing Text for Classification Purposes



Representing Text: 1. Feature Design and Extraction

- In classification by topic, a typical choice is to make the set of features coincide with the set of words that occur in the training set (unigram model, a.k.a. "bag-of-words")
- This may be preceded by (a) stop word removal and/or (b) stemming or lemmatization; (b) is meant to improve statistical robustness
- The dimensionality K of the vector space is the number of words (or stems, or lemmas) that occur at least once in the training set, and can easily be $O(10^5)$
- But each document usually contains $\ll O(10^5)$ unique words! If we indicate the absence of a word from a document by 0, this means that these vectors are usually very "sparse"
- Vector sparsity and high dimensionality are possibly the two most important characteristics that distinguish text classification from other instantiations of classification (e.g., in data mining)

Representing Text: 1. Feature Design and Extraction

- Word n-grams (i.e., sequences of n words that frequently occur in L) may be optionally added; this is usually limited to $n = 2$ (unigram+bigram model)

Word Unigrams	[A] swimmer likes swimming thus he swims A [swimmer] likes swimming thus he swims A swimmer [likes] swimming thus he swims A swimmer likes [swimming] thus he swims ...
Word Bigrams	[A swimmer] likes swimming thus he swims A [swimmer likes] swimming thus he swims A swimmer [likes swimming] thus he swims A swimmer likes [swimming thus] he swims ...

- The higher the value of n, the higher the semantic significance and the dimensionality K of the resulting representation, and the lower its statistical robustness

Representing Text: 1. Feature Design and Extraction

- An alternative to the process above is to make the set of features coincide with the set of character *n*-grams (e.g., $n \in \{3, 4, 5\}$) that occur in L; useful especially for degraded text (e.g., resulting from OCR or ASR)[1]

Character 5-grams	It was a dark and stormy night It was a dark and stormy night It was a dark and stormy night It was a dark and stormy night It was a dark and stormy night It was a dark and stormy night ...

- In order to achieve statistical robustness, all of the representations discussed so far renounce encoding word order and syntactic structure

[1] Paul McNamee, James Mayfield: Character N-Gram Tokenization for European Language Text Retrieval. *Information Retrieval* 7(1-2):73-97 (2004)

Representing Text: 2a. Feature selection

- Vectors of length $O(10^5)$ or $O(10^6)$ might result, esp. if word n-grams are used, in both "overfitting" and high computational cost;
- Feature selection (FS) has the goal of identifying the most discriminative features, so that the others may be discarded
- The "filter" approach to FS consists in measuring (via a function ξ) the discriminative power $\xi(t_k)$ of each feature t_k and retaining only the top-scoring features[2]
- For binary classification, a typical choice for ξ is mutual information, i.e.,

$$MI(t_k, c_i) = \sum_{c \in \{c_i, \overline{c}_i\}} \sum_{t \in \{t_k, \overline{t}_k\}} \Pr(t, c) \log_2 \frac{\Pr(t, c)}{\Pr(t) \Pr(c)}$$

Alternative choices are chi-square and log-odds.

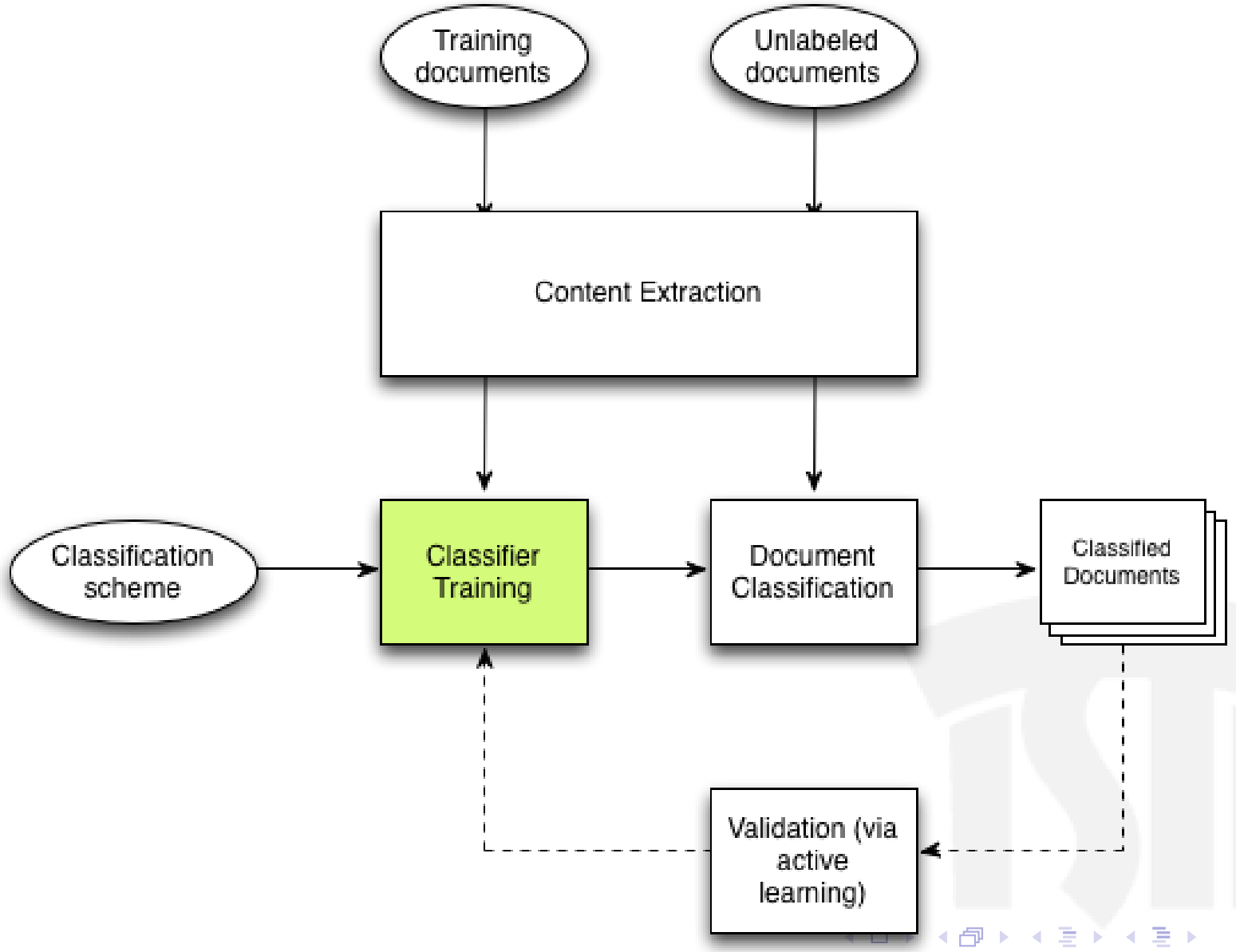
[2] Y. Yang, J. Pedersen: A Comparative Study on Feature Selection in Text Categorization. Proceedings of ICML 1997.

Representing Text: 3. Feature Weighting

Feature weighting means attributing a value x_{ik} to feature t_k in the vector $\mathbf{x}_i$ that represents document d_i: this value may be

- binary (representing presence/absence of t_k in d_i); or
- numeric (representing the importance of t_k for d_i); obtained via feature weighting functions in the following two classes:
  - unsupervised : e.g, $tf * idf$ or $BM25$,
  - supervised : e.g., $tf * MI$, $tf * \chi^2$

The Supervised Learning Approach to Classification



Supervised Learning for Binary Classification

- For binary classification, essentially any supervised learning algorithm can be used for training a classifier; popular choices include
  - Support vector machines (SVMs)
  - Boosted decision stumps
  - Logistic regression
  - Naïve Bayesian methods
  - Lazy learning methods (e.g., k-NN)
  - ...
- The "No-free-lunch principle" (Wolpert, 1996): $\approx$ there is no learning algorithm that can outperform all others in all contexts
- Implementations need to cater for
  - the very high dimensionality typical of TC
  - the sparse nature of the representations involved

Supervised Learning for Non-Binary Classification

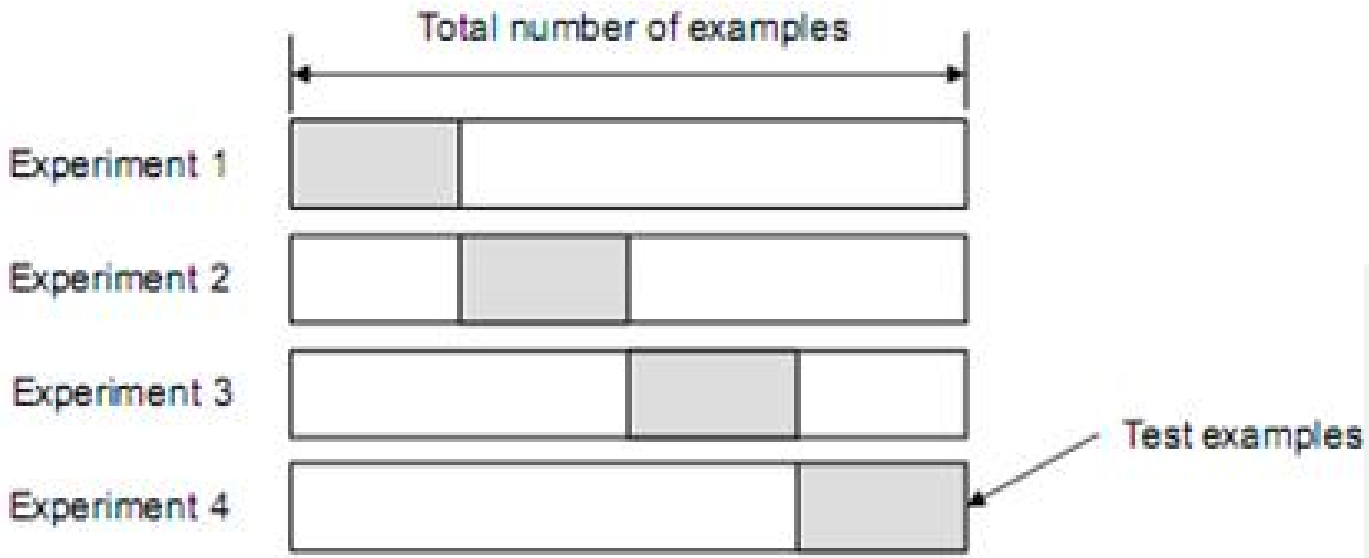
- Some learning algorithms for binary classification are "SLMC-ready"; e.g.
  - Decision trees
  - Boosted decision stumps
  - Logistic regression
  - Naive Bayesian methods
  - Lazy learning methods (e.g., k-NN)
- For other learners (notably: SVMs) to be used for SLMC classification, combinations / cascades of the binary versions need to be used[3]
- For ordinal classification, algorithms customised to OC need to be used (e.g., SVORIM, SVOREX)[4]

[3] K. Crammer and Y. Singer. On the Algorithmic Implementation of Multi-class SVMs, *Journal of Machine Learning Research*, 2001.

[4] Chu, W., Keerthi, S.: Support vector ordinal regression. *Neural Computation*, 2007.

Parameter Optimization in Supervised Learning

- The trained classifiers often depend on one or more parameters: e.g.,
  - The C parameter in soft-margin SVMs
  - The γ, r, d parameters of non-linear kernels
  - ...
- These parameters need to be optimized, e.g., via k-fold cross-validation on the training set



Text Classification

Evaluating a Classifier

- Two important aspects in the evaluation of a classifier are efficiency and effectiveness
- Efficiency refers to the consumption of computational resources, and has two aspects
  - Training efficiency (also includes time devoted to performing feature selection and parameter optimization)
  - Classification efficiency; usually considered more important than training efficiency, since classifier training is carried out (a) offline and (b) only once
- For evaluating a text classifier it is good practice to consider both training costs and classification costs

Effectiveness

- Effectiveness (a.k.a. accuracy) refers to how frequently classification decisions taken by a classifier are "correct"
- Usually considered more important than efficiency, since accuracy issues "are there to stay"
- Effectiveness tests are carried out on one or more datasets meant to simulate operational conditions of use
- The main pillar of effectiveness testing is the evaluation measure we use

Evaluation Measures for Classification

- Each type of classification (binary/SLMC/MLMC/ordinal) and mode of classification (hard/soft) requires its own measure
- For binary (hard) classification, given the contingency table Ω

		true	
		YES	NO
pred	YES	TP	FP
	NO	FN	TN

the standard measure is F_1, the harmonic mean of precision ($\pi = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$) and recall ($\rho = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$), i.e.,

$$F_1 = \begin{cases} \frac{\pi\rho}{\pi+\rho} = \frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}} & \text{if } \mathrm{TP}+\mathrm{FP}+\mathrm{FN} > 0 \\ 1 & \text{if } \mathrm{TP} = \mathrm{FP} = \mathrm{FN} = 0 \end{cases}$$

- F_1 is robust to the presence of imbalance in the test set

Evaluation Measures for Classification (cont'd)

- For multi-label multi-class classification, F_1 must be averaged across the classes, according to
  1. micro-averaging: compute F_1 from the "collective" contingency table obtained by summing cells

		true	
		YES	NO
predicted	YES	$\sum_{c_i \in \mathcal{C}} \mathrm{TP}_i$	$\sum_{c_i \in \mathcal{C}} FP_i$
	NO	$\sum_{c_i \in \mathcal{C}} \mathrm{FN}_i$	$\sum_{c_i \in \mathcal{C}} \mathrm{TN}_i$

  2. macro-averaging: compute $F_1(c_i)$ for all $c_i \in \mathcal{C}$ and then average
- Micro usually gives higher scores than macro ...

Evaluation Measures for Classification (cont'd)

- For single-label multi-class classification, the most widely used measure is ("vanilla") accuracy

$$A = \frac{\sum_{c_i \in \mathcal{C}} \Omega_{ii}}{|U|}$$

where Ω_{ij} is the number of documents in c_i which are predicted to be in c_j

		true		
		c_1	...	$c_{\lvert\mathcal{C}\rvert}$
pred	c_1	Ω_{11}	...	$\Omega_{1\lvert\mathcal{C}\rvert}$
	...	...	...	...
	$c_{\lvert\mathcal{C}\rvert}$	$\Omega_{\lvert\mathcal{C}\rvert 1}$	...	$\Omega_{\lvert\mathcal{C}\rvert\lvert\mathcal{C}\rvert}$

- Macro-averaged F_1 is a possible alternative

Evaluation Measures for Classification (cont'd)

- For ordinal classification, the measure must acknowledge that different errors may have different weight; the most widely used is macroaveraged mean absolute error, i.e.,

$$MAE^M(h, U) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{|U_i|} \sum_{\mathbf{x}_j \in U_i} |h(\mathbf{x}_j) - y_i)|$$

- For soft classification, measures from the tradition of ad hoc retrieval are used. E.g., for soft single-label multi-class classification, mean reciprocal ranking can be used, i.e.,

$$MRR(h, U) = \frac{1}{|U|} \sum_{\mathbf{x}_j \in U} \frac{1}{r_h(y_i)}$$

Some "Classic" Datasets for Evaluating Text Classification

	Total examples	Training examples	Test examples	Classes	Hierarchical	Language	Type
Reuters-21578	$\approx$ 13,000	$\approx$ 9,600	$\approx$ 3,200	115	No	EN	MLMC
RCV1-v2	$\approx$ 800,000	$\approx$ 20,000	$\approx$ 780,000	99	Yes	EN	MLMC
20Newsgroups	$\approx$ 20,000	—	—	20	Yes	EN	MLMC
OHSUMED-S	$\approx$ 16,000	$\approx$ 12,500	$\approx$ 3,500	97	Yes	EN	MLMC
TripAdvisor-15763	$\approx$ 15,700	$\approx$ 10,500	$\approx$ 5,200	5	No	EN	Ordinal
Amazon-83713	$\approx$ 83,700	$\approx$ 20,000	$\approx$ 63,700	5	No	EN	Ordinal

Want to Experiment with Text Classification?

- Several publicly available environments where to play with text preprocessing routines, feature selection functions, feature weighting functions, learning algorithms, etc. E.g.,
  - SCIKIT-LEARN (http://scikit-learn.org/): Python-based, features various classification, regression and clustering algorithms including SVMs, random forests, gradient boosting, k-means (...), and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.
  - WEKA (https://www.cs.waikato.ac.nz/ml/weka/): Java-based, features various algorithms for data analysis and predictive modeling.

Text Classification

Advanced Topics (hints)

- Hierarchical classification
  - When the codeframe has a hierarchical nature
- Hypertext classification (an application of "Relational Learning")
  - When the items are hypertextual (e.g., Web pages)
- Cost-sensitive classification
  - When false positives and false negatives are not equally bad mistakes
- Semi-supervised classification
  - When the classifier is trained using a combination of labelled and unlabelled documents
- Transductive classification
  - When at training time we have all the unlabelled texts that need classifying
- Semi-automated text classification (aka "technology-assisted review")
  - Optimizing the work of human assessors that need to review the results of automated classification
- Active learning for classification
  - When the items to label for training purposes are suggested by the system

Further Reading

- General:
  - C. Aggarwal and C. Zhai: A Survey of Text Classification Algorithms. In C. Aggarwal and C. Zhai (eds.), *Mining Text Data*, pp. 163–222, Springer, 2012.
  - C. Aggarwal: Chapters 5–7 of *Machine Learning for Text*, Springer, 2018.
- Supervised learning:
  - K. Murphy: *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.
  - T. Hastie, R. Tibshirani, J. Friedman: *The Elements of Statistical Learning*, 2nd Edition. Springer, 2009.
- Evaluating the effectiveness of text classifiers:
  - N. Japkowicz and M. Shah: *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, 2011.

Word Embeddings for Text Classification

Alejandro Moreo & Fabrizio Sebastiani

ESSIR2022, 18-22 July, Lisbon, PT

Word Embeddings

From Word Space Model
to
Neural Embeddings

Word Space Model

Word Similarity

Dense Representations

Word2Vec

Vector Space Model

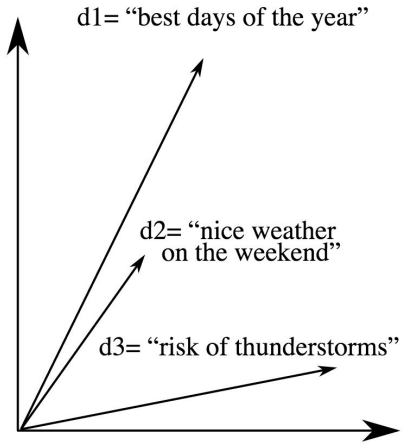
The *Vector Space Model* (VSM) is a machine-friendly **representation** for text.

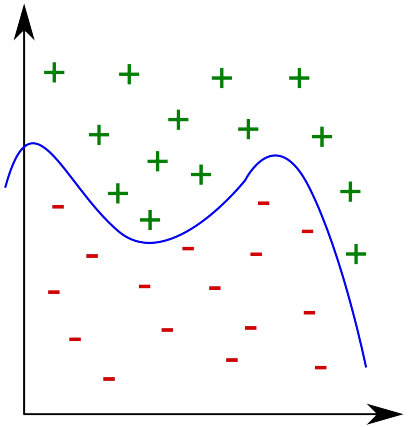
One-hot representation: every **word** is an orthogonal dimension

$$v(\text{cat}) = [0, 0, \ldots, \ldots, \mathbf{1}, 0]$$

Documents are *sums of one-hot* vectors (weighted by term importance, e.g., TFIDF)

$$v(d_1) = [w_1, w_2, \ldots, \ldots, w_{n-1}, w_n]$$





Sparse representations

d_1 = 'you played a game'
d_2 = 'you played a match'
d_3 = 'you played a trumpet'

$v(d_1)$ = [0, $w_{played,d1}$, $w_{game,d1}$, 0 , 0, … , 0, 0 , 0]
$v(d_2)$ = [0, $w_{played,d2}$, 0 , $w_{match,d2}$, 0, … , 0, 0 , 0]
$v(d_3)$ = [0, $w_{played,d3}$, 0 , 0 , 0, … , 0, $w_{trumpet,d3}$, 0]

Semantic similarity between features (game~match) is **not captured**:

$$sim(v(d_1), v(d_2)) \sim sim(v(d_1), v(d_3)) \sim sim(v(d_2), v(d_3))$$

Similar words are as close to each other as words with **different meanings**

Modeling word similarity

How do we model that *game* and *match* are related terms but *trumpet* is not?

Using linguistic resources? it requires a lot of human work to build them.

Observation: co-occurring words are semantically related.

Pisa	*is a*	*province*	*of*	*Tuscany*
Red	*is a*	*color*	*of the*	*rainbow*
Wheels	*are*	*components*	*of the*	*bicycle*
**Red*	*is a*	*province*	*of the*	*bicycle*

Exploit this propriety of language following the *distributional hypothesis*.

Distributional hypothesis

“You shall know a word by the company it keeps!” Firth (1957)

Distributional hypothesis: the meaning of a word is determined by the contexts in which it is used (Wittgestein).

*Yesterday we had **bim** at the restaurant.*
*I remember my mother cooking me **bim** for lunch.*
*I don't like **bim**, it's too sweet for my taste.*
*I like to dunk a piece of **bim** in my morning coffee.*

Word-Context matrix (or “Context counting”)

Distributional Semantic Model: capture the words semantic

Word-context matrix: is a $|F| \cdot |F|$ matrix X that **counts** the frequencies of co-occurrence of **words** in a collection of **contexts**.

You cook the cake twenty minutes in the oven at 220 C.
I broke a tire hitting a curb, I changed the tire.

$\text{Context}_{-2,+2}$('cake') = {['cook','the','twenty', 'minutes']}
$\text{Context}_{-2,+2}$('tire') = {['broke','a','hitting', 'a'], ['changed', 'the']}

Word-Context matrix

		Context words						
		…	cook	eat	…	changed	broke	…
Words	cake	…	10	20	…	0	0	…
	steak	…	12	**22**	…	0	0	…
	bim	…	7	10	…	0	0	…
	engine	…	0	0	…	3	10	…
	tire	…	0	0	…	10	1	…
	…	…	…	…	…	…	…	…

$Words \equiv Context\ words$

Context vector

Similar words have *similar rows*

Rows of the Word-Context are **non-orthogonal**

We can use them as *v(f)* to build a “more semantic” representation of documents

$$v(d) = \sum_{f \in d} w_{fd} v(f)$$

yet such vectors are still **high dimensional** and largely **sparse**.

Dense representations

Project feature vectors *v(f)* into a *low-dimensional* space R^k, with $k \ll |F|$

Embeddings: *dense* representations in a *continuous low dimensional space*

$$Embed: R^{|F|} \rightarrow R^k \qquad Embed(v(f)) = e(f)$$

We force features to share dimensions on a **reduced dense space**

↓

Let's group/align/project them by their syntactic/semantic similarities!

SVD

Singular Value Decomposition is a decomposition method of a matrix X of size $m \cdot n$ into three matrices $U\Sigma V^*$, where:

U is an orthonormal matrix of size $m \cdot n$

Σ is a diagonal matrix of size $n \cdot n$, with values $\sigma_1, \sigma_2 \dots \sigma_n$

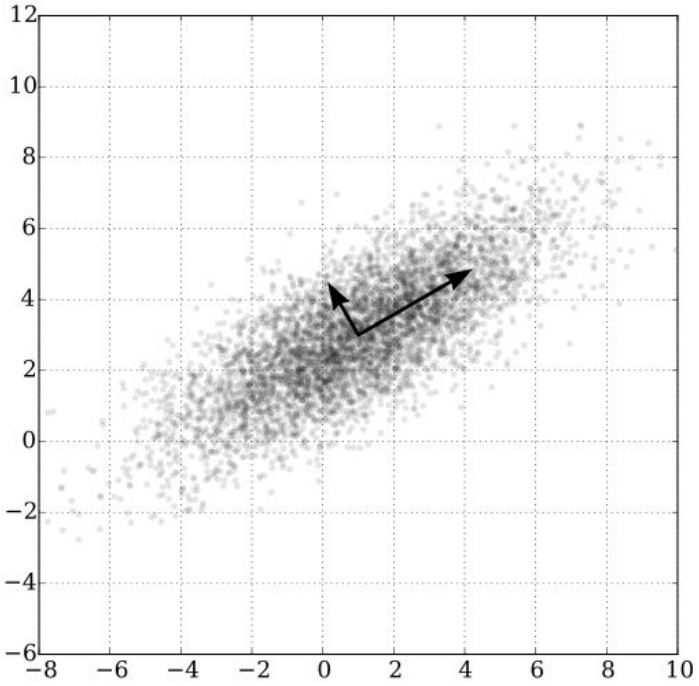
V is an orthonormal matrix of size $n \cdot n$, V^* is its conjugate transpose

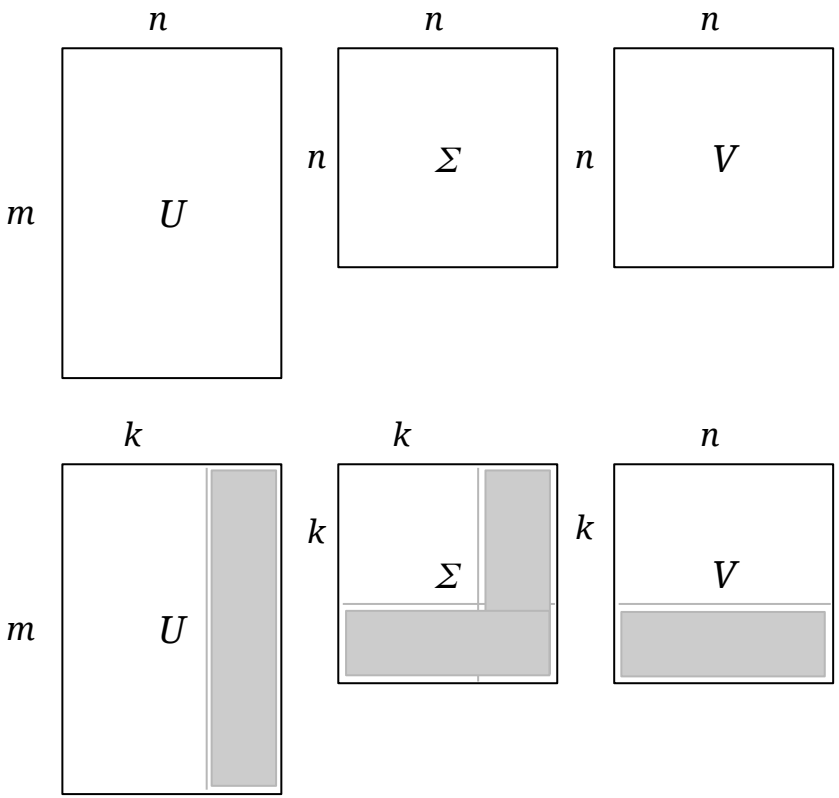
$\sigma_1, \sigma_2 \dots \sigma_n$ of Σ are the singular values of X, sorted by decreasing magnitude.

Keeping the top k values gives a least-square approximation of X

Rows of U_k of size $m \cdot k$ are the dense representations of the features

SVD





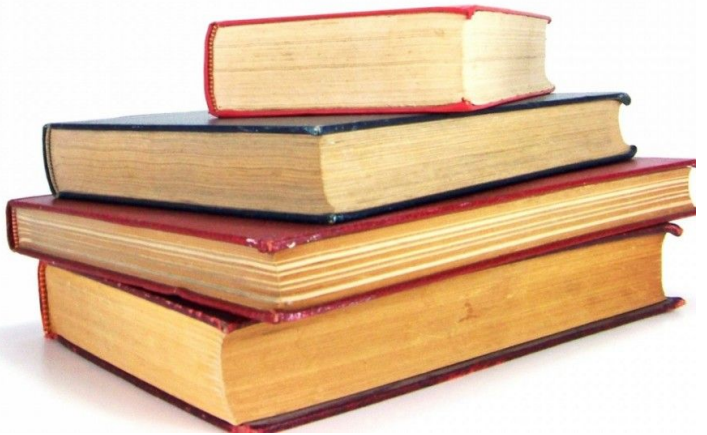
By Nicoguaro (Own work) [CC BY 4.0], via Wikimedia Commons

Neural models

We can use a neural network to model the distributional hypothesis, providing it with examples of use of terms in context (**a lot of text**).

Recent complex, and **large**, neural models have shown that language models not only can capture knowledge about the use of language, but also common knowledge about the world.

Examples: GloVe, **word2vec**, MUSE, ...



“Context Prediction” methods

What are the missing words?

machine learning uses _____ data to learn a model …
the distance between _____ and the Sun is …

Task: Predicting the missing word given a context

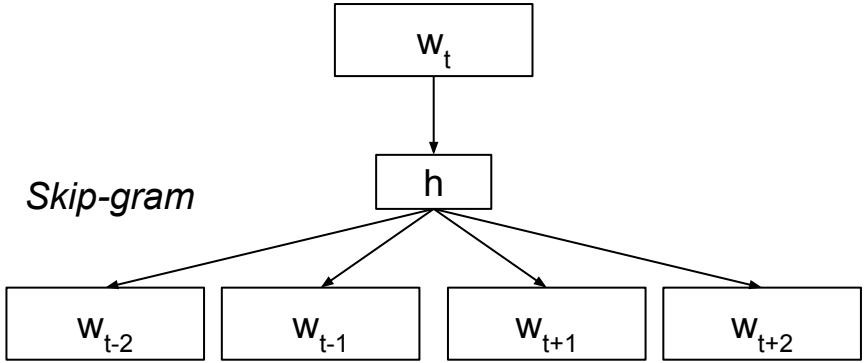
We just need (a lot of) text as the training data…
no need for additional human-labeled data! (**unsupervised**)

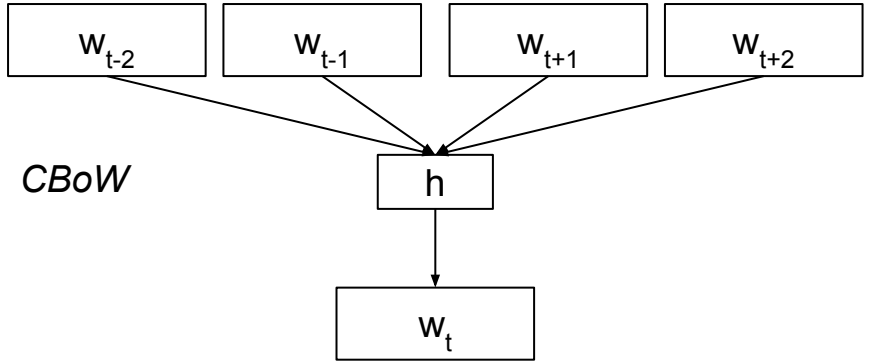
The context window size can change, longer windows capture **semantic**, shorter capture **syntax** (typically 5-10 words).

Word2Vec (Mikolov, 2013)

Skip-gram and CBoW models of Word2Vec define tasks of **predicting** a *context from a word* (Skip-gram) or a *word from its context* (CBoW).

They are both implemented as a two-layer linear **neural networks** in which input and output words one-hot representations which are encoded into a dense representation of smaller dimensionality.





“Counting” vs “Prediction”: which is better?

Levy and Goldberg (2014) proved that W2V SGNS implicitly computes a factorization of a variant of X (called shifted PPMI).

Levy, Goldberg and Dagan (2015) comparison of SVD, CBoW, SGNS, GloVe.

- Results: **no clear winner**
- **Hyperparameters** play a relevant role
- Both SVD and W2V performed well on most tasks
- **W2V** has **lower computational** cost in time and memory (bag of “tricks”)

Computing embeddings

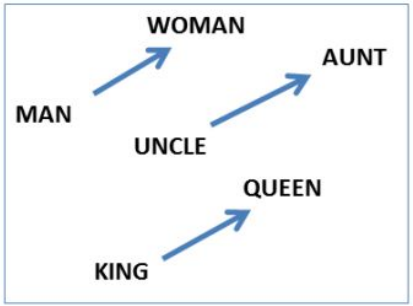
Gensim provides an efficient and detailed implementation.

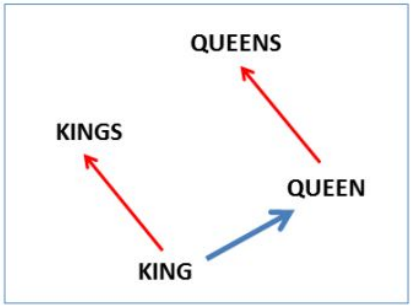
```
sentences = [['this','is','a','sentence'],
             ['this','is','another','sentence']]

from gensim.models import Word2Vec
model = Word2Vec(sentences)
```

This is a clean implementation of skip-grams using pytorch.

Testing embeddings

Testing if embeddings capture syntactic/semantic properties.





Analogy test:

Paris stands to *France* as *Rome* stands to _____?

$$e('France') - e('Paris') \sim e('Italy') - e('Rome')$$

$$e('France') - e('Paris') + e('Rome') \sim e('Italy')$$

$$d = \arg\max_x \frac{(e(b)-e(a)+e(c))^T e(x)}{||e(b)-e(a)+e(c)||}$$

Embeddings in neural networks

An embedding layer in neural networks is typically the **first layer** of the net.

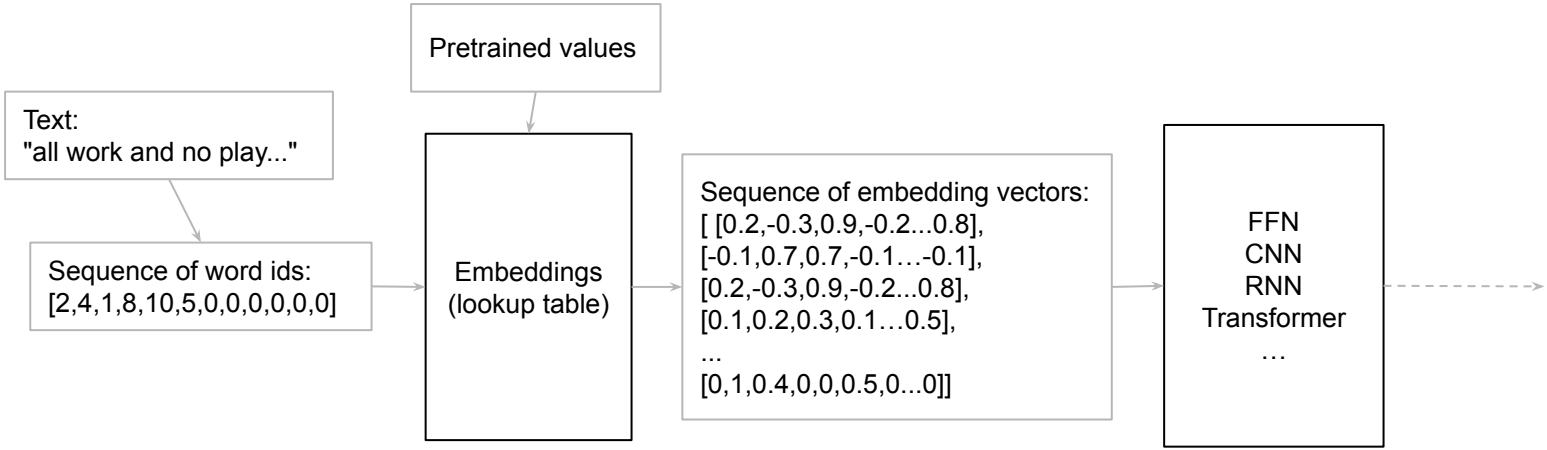
A matrix W of size $|F| \cdot n$, where n is the size of the **embedding space**.

It maps words to dense representations (**lookup table**).

Alternatively: It can be initialized with random weights, and learn for a task.

During learning, weights can be kept **fixed** (only when using pretrained embeddings) or be **updated**, to adapt embeddings to the task at hand.

Embeddings in neural networks



Word Embeddings in Text Classification

From Neural Embeddings
to
document labels

Word Space Model

Word Similarity

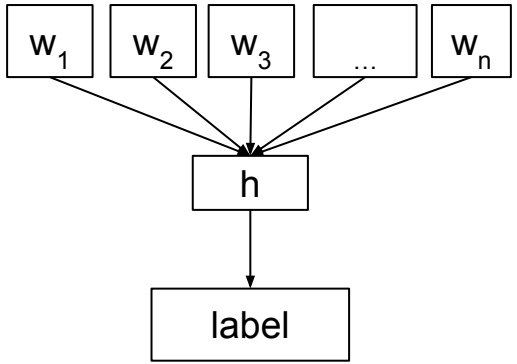
Dense Representations

Word2Vec

FastText (Joulin et al., 2016)

FastText extends the W2V

- Context: the entire document
- Predict: the document label



Model ngrams: resiliency to OOV. E.g., “goodbye”

"<go", "goo","ood", "odb", "dby", "bye", "ye>"

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

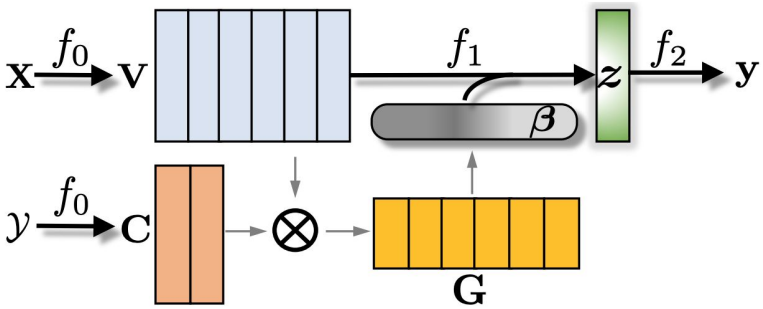
LEAM (Wang et al., 2018)

Label Embedding Attentive Model

Jointly models:

- Word embeddings (**v**)
- Label embeddings (**c**)

Attention mechanism (**G**): weights of words



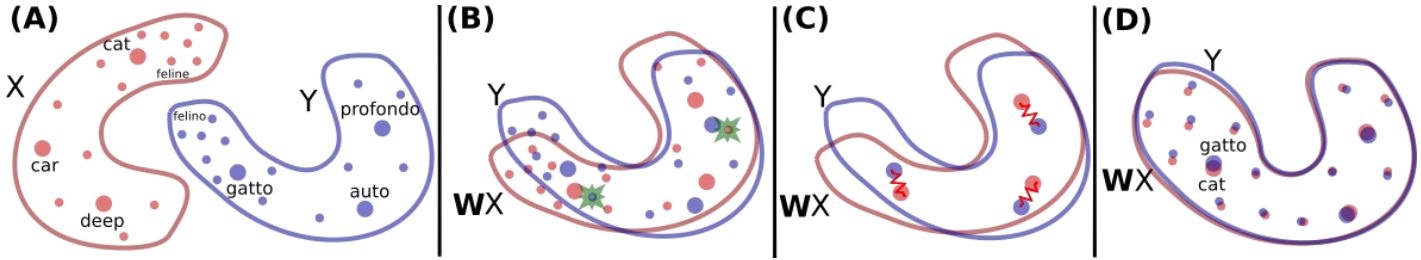
[image source]

Wang, G., Li, C., Wang, W., Zhang, Y., Shen, D., Zhang, X., ... & Carin, L. (2018). Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174*.

Multilingual Embeddings (Conneau et al., 2017)

MUSE (Multilingual Unsupervised and Supervised Embeddings)

- *Supervised:* Procrustes alignment using *bilingual dictionaries*
- *Unsupervised*: using *adversarial learning* to fool a discriminator in distinguishing the source language



Conneau, A., Lample, G., Ranzato, M. A., Denoyer, L., & Jégou, H. (2017). Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Word Class Embeddings (Moreo et al., 2021)

- Word embeddings capture **word-word** correlations (*general)*
- *Levy and Goldberg 2014* proved:

$$\mathbf{e}_i = \psi([\eta(w_i, c_1), \ldots, \eta(w_i, c_{|V|})])$$

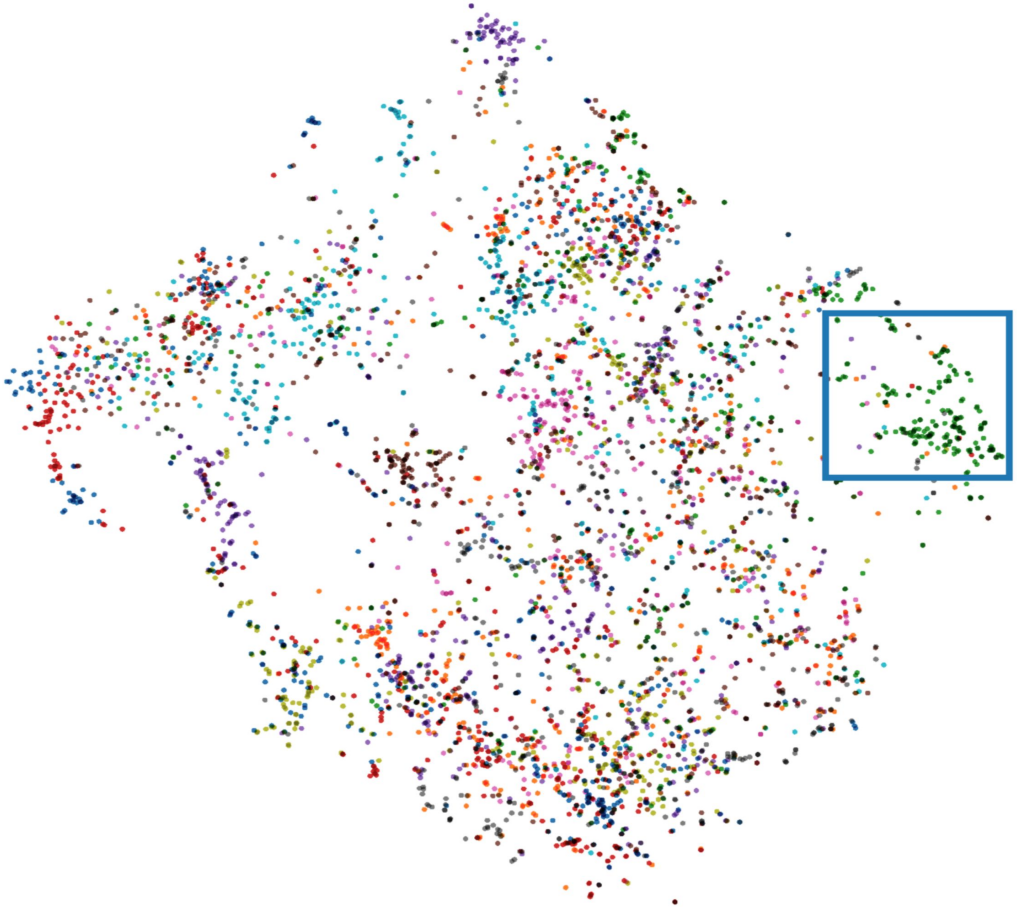
- WCE: **word-class** correlations (*specific*):

$$\mathbf{e}_i = \psi([\eta(w_i, y_1), \ldots, \eta(w_i, y_m)])$$

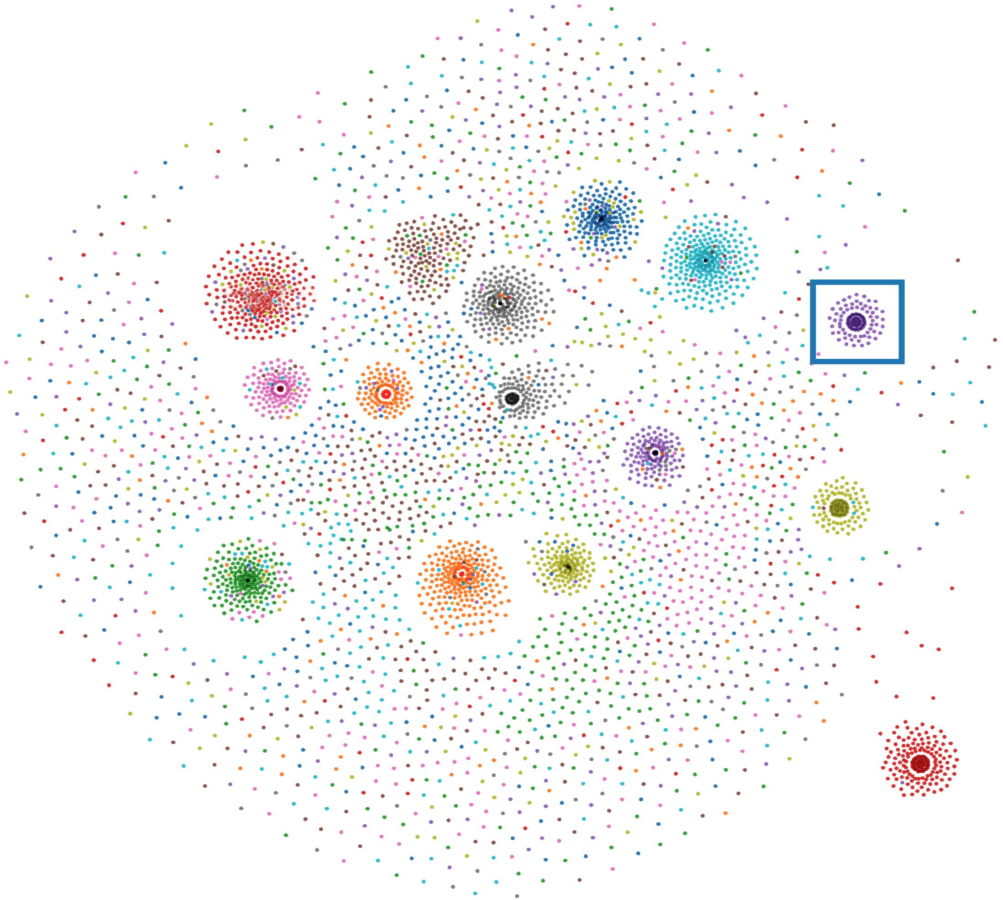
$$\mathbf{E} := \mathrm{PCA}_r(\mathbf{X}_1^\top \mathbf{Y})$$

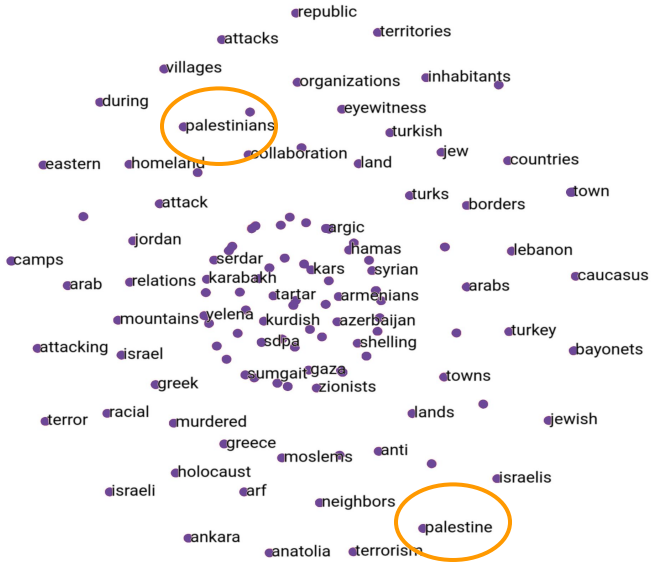
Moreo, A., Esuli, A., & Sebastiani, F. (2021). Word-class embeddings for multiclass text classification. *Data Mining and Knowledge Discovery*, *35*(3), 911-963.

Word2Vec Embeddings

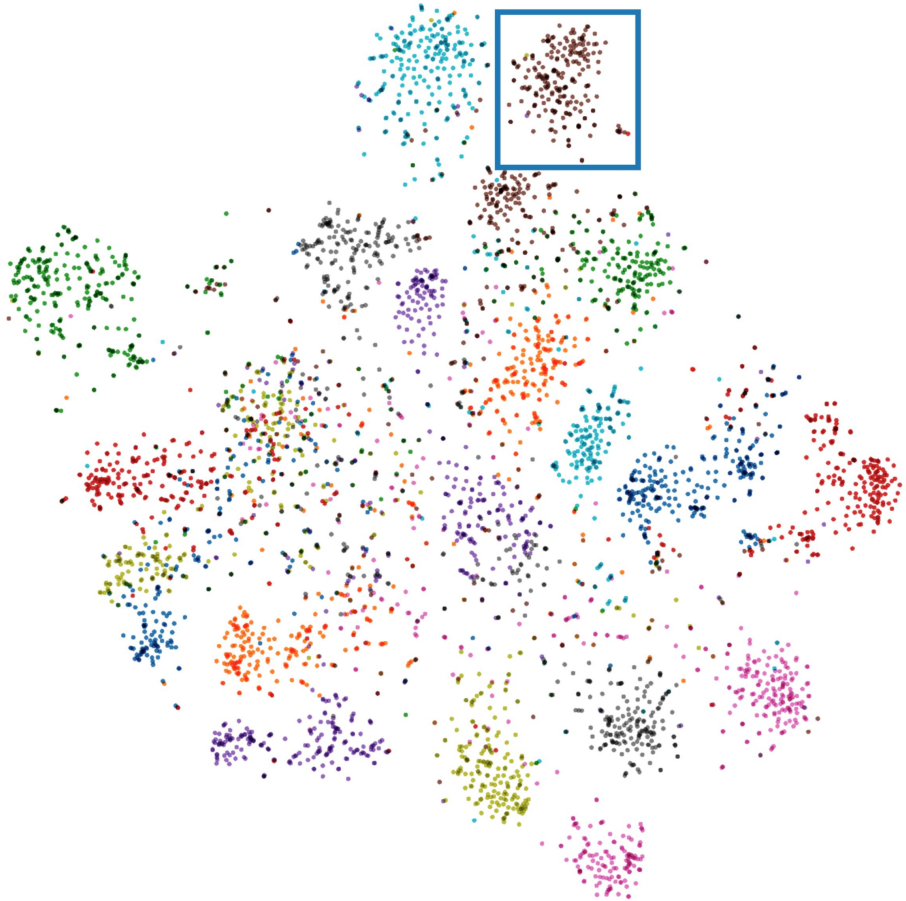


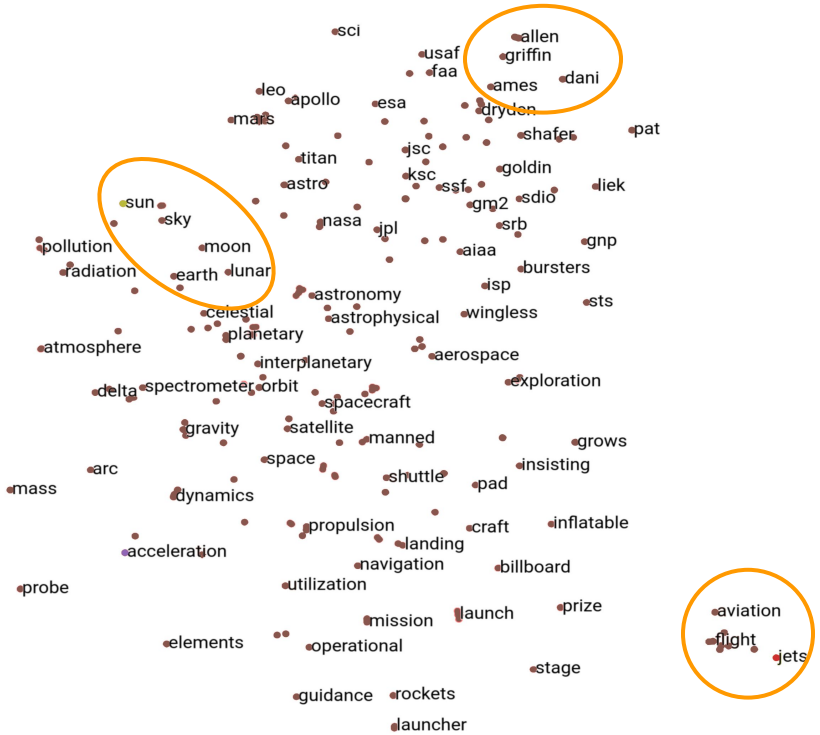
Word-Class Embeddings





Word2Vec + Word-Class Embeddings





Current Trend

The “traditional” deep learning architectures for text classification:

- Feed-forward: e.g., fastText
- Recurrent Neural Networks: e.g., LSTM
- Convolutional Neural Networks: e.g., character-based
- “**Static**” word embeddings + a deep neural network

The “modern” architectures:

- Transformers: e.g., BERT, GPT2, XLNet
- Huge models trained on huge data for general language modelling tasks
- Already encode a good model for language understanding
- “**Contextualized**” word embeddings + a linear layer

A shift of paradigm

Traditional machine learning:

- A knowledge engineer / experts decide the features (90%)
- A general model tries to make sense of them (10%)

Static word embeddings:

- Words are represented offline, encode general use (50%)
- A deep neural network is trained on top (50%)

Transformers:

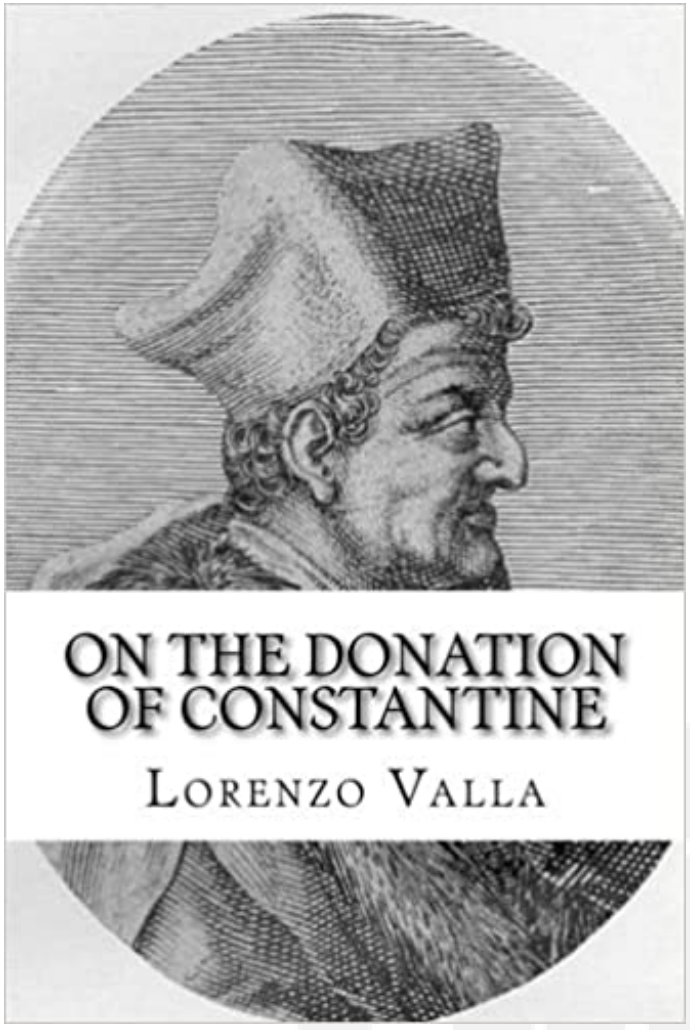
- A deep model is trained offline, encode general use (90%)
- A simple linear layer adapts the output to the task at hand (10%)

Questions?

Part II : Authorship Analysis

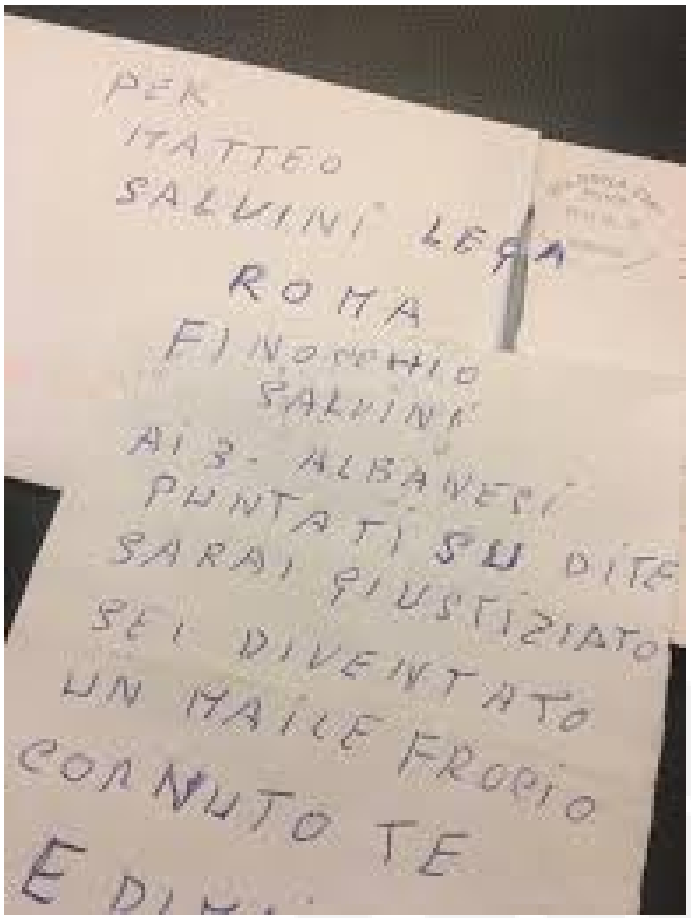
Spotting fake texts

- Can we spot a fake text?
- Different notions of what a fake text is:
  1. A text that reports false facts, sometimes fabricated ones, usually presented as being factually accurate (as in "fake news")
  2. A text whose author (a forger) pretends to be a different author
- The latter is the meaning we will be looking at



Spotting fake texts

- On Jan 1, 2018 Italian then-Prime Minister M. Salvini publicized this anonymous letter that he allegedly received
- The letter contained several threats, and looked like it was written (in uncertain Italian) by an Albanian



Spotting fake texts

- Three days later, Albanian sociolinguist E. Shkreli (UofBologna), argued that the message was a forgery, since
  - while it showed poor knowledge of Italian, it did not contain typical mistakes (with articles, or double consonants, or ...) that L1 Albanian speakers make when writing in L2 Italian
  - it contained mistakes ("ai" instead of "hai") that Italians with low proficiency in written Italian typically make;
  - it contained idioms ("puntati su di te") that are "very Italian", and unnatural for Albanians.

Spotting fake texts

- The above is an attempt at Native Language Identification, the task of identifying the L1 of the author of a text written in a language L2
- NLI is based on the fact that learners of an L2 display a tendency to transfer forms and meanings of their L1 linguistic background to L2 (language transfer, aka L! interference)
- Q: Can we automate NLI?
- A: Yes.

MENU | CERCA

la Repubblica

HOME CRONACA SPORT FOTO RISTORANTI ANNUNCI LOCALI

Bologna, la prof albanese corregge Salvini: "Quelle minacce non sono nostre, ecco perché"

Matteo Salvini
@salviniofficial
Home
Informazioni
Post
Foto

Il ministro dell'Interno aveva pubblicato sui social una lettera sgrammaticata stranieri. Ma la docente, che insegna nel dipartimento di Lingue dell'Alma Mat facciamo quel tipo di errori grammaticali"

di ROSARIO DI RAIMONDO

ABBONATI A Rep: 05 dicembre 2018

BOLOGNA - Quella pesante lettera di insulti e minacce non l'ha scritta un albanese. **Enkelejda Shkreli** ne è certa. Non perché sia una detective. Ma perché è albanese, parla perfettamente italiano ed è docente a contratto del dipartimento di Lingue dell'Università di Bologna. Ma soprattutto perché conosce il suo popolo e gli errori grammaticali che commette quando parla o scrive nella nostra lingua.

(Computational) Authorship Analysis

- NLI is one example of Authorship Analysis, the task of predicting / guessing / inferring the characteristics of the author of a text of unknown or disputed authorship
- AA: Traditionally carried out by linguists and philologists, via
  1. a linguistic analysis of the characteristics present in the disputed text (e.g., word "satrap" + poor quality of Latin in the *Donation of Constantine*)
  2. an extralinguistic analysis of concepts expressed and facts described in the text, and the likelihood that a certain author could express and describe them
- Computational Authorship Analysis is the attempt to perform authorship analysis by computerized means, and usually rests only on linguistic (and no extralinguistic) analysis

(Computational) Authorship Analysis

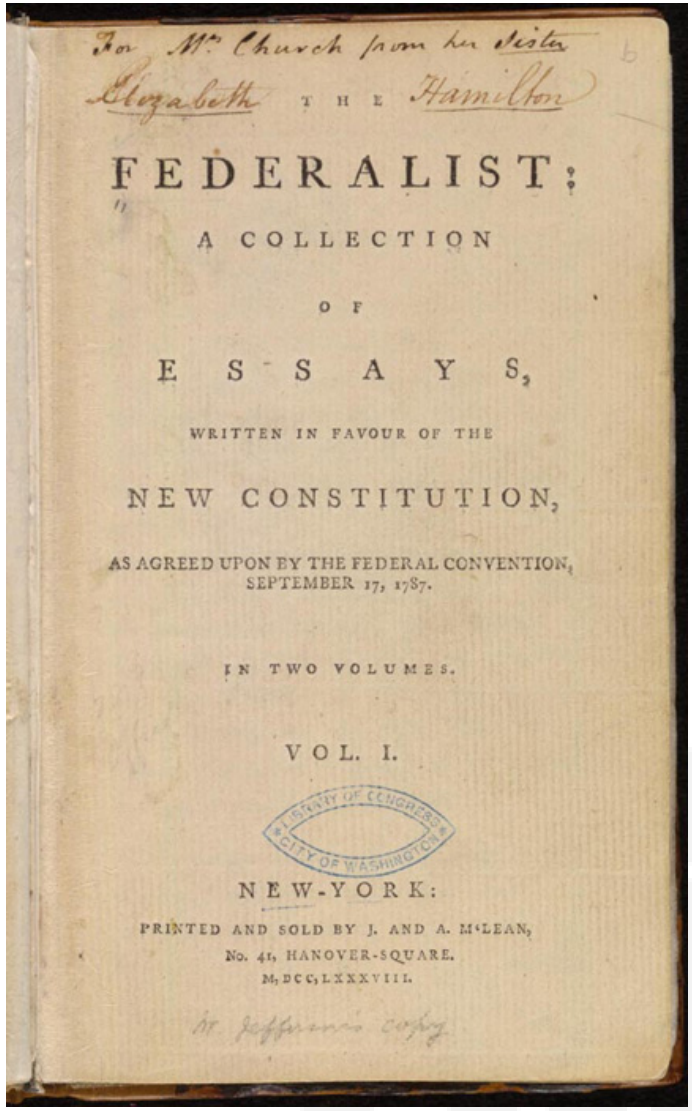
- Various sub-tasks of (Computational) Authorship Analysis; e.g.,
  - tasks dealing with inferring the identity of the author; e.g.,
    - Authorship Attribution (AA), i.e., predicting who, among a set of k candidate authors, is the most likely author of the text;
    - Authorship Verification (AV), i.e., predicting if a certain candidate author is or not the author of the text;
    - Same-Authorship Verification (SAV), i.e., predicting whether two candidate texts d' and d'' are by the same author or not;
  - tasks dealing with inferring other characteristics of the author; e.g.,
    - Native Language Identification (NLI), i.e., predicting the most likely native language (L1) of the author of the text, out of a set of k candidate native languages
    - Gender Identification (GI), i.e., predicting whether the text was written by a woman or a man;
    - Bot Detection (BD), i.e., predicting whether the text (usually: a social media post) was written by a human or a "bot".
- Here we will mostly deal with authorship verification.

(Computational) Authorship Analysis

- Major applications of authorship analysis include
  - Cybersecurity, i.e., the prevention of crimes that could be committed with the help of digital means
  - Computational forensics, i.e., the digital analysis of evidence from crimes that have already been committed.
    - forensic linguistics can do for crimes involving language, such as threats, blackmail, and extortion, what DNA has done for violent crimes.
  - Philology: discussed below

Computational Authorship Analysis and Philology

- Applications of computational authorship analysis to texts of literary or historical value include
  - Mosteller & Wallace 1964 : Who among the "Publius" collective wrote the *Federalist* papers?
  - Italia and Canettieri 2013 : is "Montale's Posthumous Diary" authentic?
  - Tuccinardi 2017: is "Pliny the Younger's letter to Trajan on the Christians" authentic?
  - (Various authors) 2017 : Who is Elena Ferrante?
- Computational authorship analysis is not meant to replace the work of philologists, but to provide them with new tools and hypotheses


For Mr. Church from his sister Elizabeth

THE Hamilton

FEDERALIST:

A COLLECTION

OF

ESSAYS,

WRITTEN IN FAVOUR OF THE

NEW CONSTITUTION,

AS AGREED UPON BY THE FEDERAL CONVENTION, SEPTEMBER 17, 1787.

IN TWO VOLUMES.

VOL. I.

LIBRARY OF CONGRESS CITY OF WASHINGTON

NEW-YORK:

PRINTED AND SOLD BY J. AND A. M'LEAN,

No. 41, HANOVER-SQUARE.

M, DCC, LXXXVIII.

Mr. Jefferson's copy

Authorship Analysis and Stylometry

- We tackle authorship analysis as a text classification task
- What differentiates classification by author from classification by topic is the choice of features (each dimension of classification is characterized by its choice of features)
- As usual, if the features have been chosen well, data items belonging to the same class (= author) will be close to each other in the vector space

Wincenty Lutosławski (1863–1954)

Authorship Analysis and Stylometry

- As usual, the choice of the "right" features is thus an art that the designer of ML-based classifiers must master
- In their choice of features, the designers of authorship analysis usually look at stylometry, the discipline that studies linguistic style via quantitative means
- Instance of the "evidential paradigm" postulated by Carlo Ginzburg in his essay "Clues"

Wincenty Lutosławski (1863–1954)

Authorship Analysis and Stylometry

- Typical stylometry-inspired features used in authorship analysis are
  - punctuation symbols
  - word lengths
  - sentence lengths
  - token/word ratio
  - function words
  - POS tags and POS-grams
  - rhythmic features
  - ...
- The assumption is that the frequency of use of these features tends to fall outside the conscious control of the author, and is thus
  - author-invariant (a "digital fingerprint")
  - hard to copy for a would-be forger

Wincenty Lutosławski (1863–1954)

Authorship Analysis and Stylometry

- Stylometry-inspired features would be useless in classification by topic, and content-bearing features are useless (actually: confounding) in classification by author
- As a result, content-bearing features are often discarded in authorship analysis
- While the choice of feature varies across different dimensions of classification, other aspects (e.g., feature selection functions, learning algorithms, evaluation measures) do not

Wincenty Lutosławski (1863–1954)

A case study: Dante's "Epistle to Cangrande"

- A case study:

  Were the two parts of Dante's "Epistle to Cangrande" actually written by Dante, or were they written by a forger?

- A long-standing problem in philology
- We tackle it via (computational) authorship verification, the task of predicting if a candidate author a^* is or is not the author of a text d of unknown paternity

A case study: Dante's "Epistle to Cangrande"

- Solved as a binary text classification problem, using "stylometric" features (i.e., stylistic traits)
- "One vs. the rest" classifier trained on texts by author a^* (positive examples) and on texts by other authors $\mathcal{A} = \{a_1, ..., a_n\}$ (negative examples)
- In order to do so we assemble two corpora of Latin texts (one for each part) written by Dante's coeval authors

Authorship Verification

- Our AV system hypothesized that both parts of the Epistle were written by a forger
- This hypothesis is corroborated by high accuracy results that the same system has obtained on texts of known authorship

	LOO Ep13(1)	LOO Ep13(2)
TP	**11**	**1**
FP	0	2
FN	1	1
TN	**282**	**26**
Total	294	30

References

- Carlo Ginzburg. 1989. Clues: Roots of an Evidential Paradigm. In *Clues, Myths, and the Historical Method: Works of Carlo Ginzburg*. The Johns Hopkins University Press, Baltimore, US, 96–214.
- Patrick Juola. 2006. Authorship Attribution. *Foundations and Trends in Information Retrieval* 1, 3 (2006), 233–334. DOI: http://dx.doi.org/10.1561/1500000005
- Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2009. Computational methods in authorship attribution. *Journal of the American Society for Information Science and Technology* 60, 1 (2009), 9–26. DOI: http://dx.doi.org/10.1002/asi.20961
- Efstathios Stamatatos. 2009. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology* 60, 3 (2009), 538–556. DOI:http://dx.doi.org/10.1002/asi.21001
- Efstathios Stamatatos. 2016. Authorship Verification: A Review of Recent Advances. *Research in Computing Science* 123 (2016), 9–25.

Part II :
Multilingual Text Classification

Multilingual Text Classification Made Easy (and Effective too)

Alejandro Moreo

Joint work with

Andrea Esuli, Andrea Pedrotti, Fabrizio Sebastiani

Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
56124 Pisa, IT
E-mail: alejandro.moreo@isti.cnr.it

ESSIR2022
Lisbon, Portugal – Jul 18-22, 2022

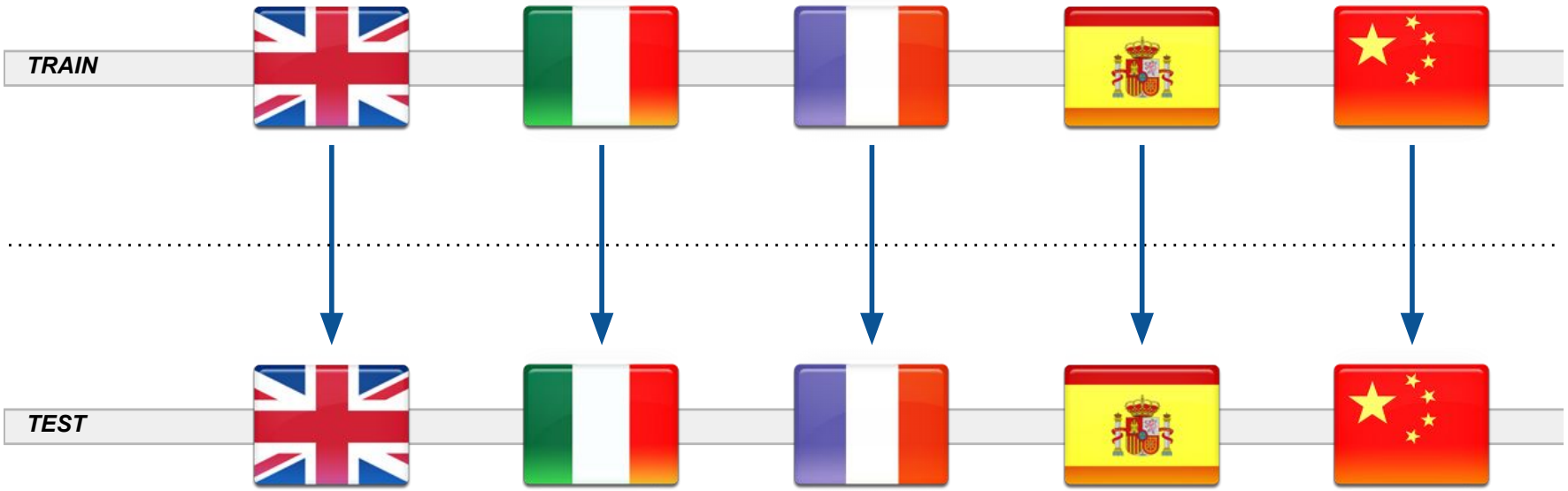
What is this talk about?

- Multilingual text classification (by topic, by sentiment, ...) ...
- ... and simple+effective ways to tackle it, via ...
  - classifier ensembles
  - transfer learning
  - word embeddings

Multilingual Text Classification

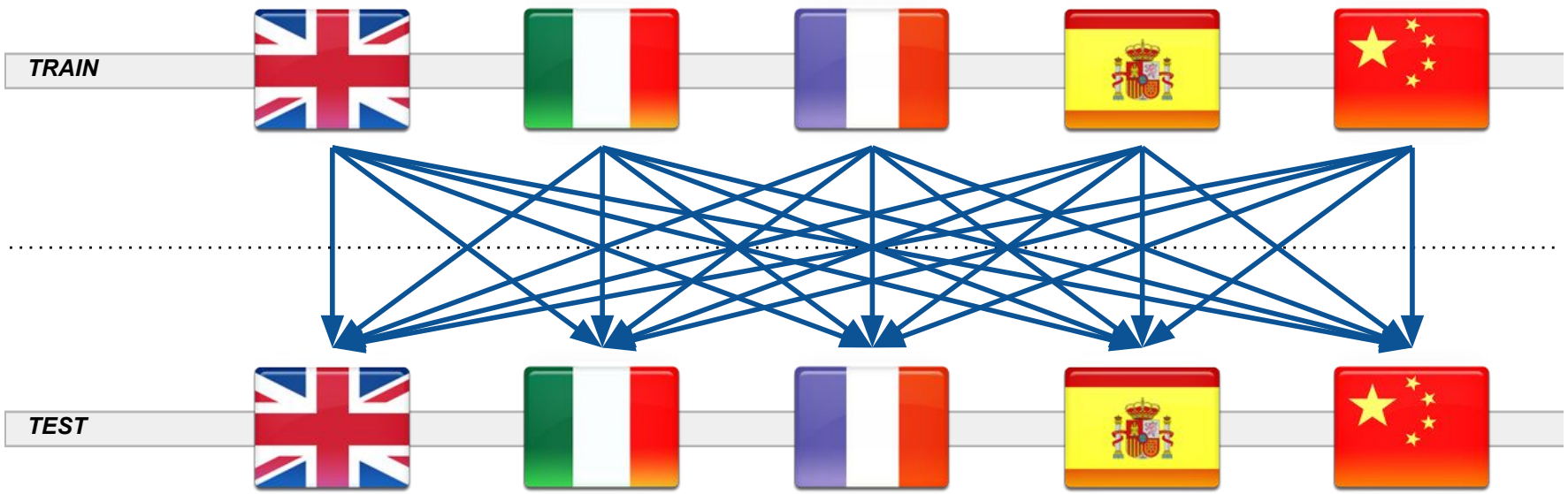
- Each document d written in one of a finite set $\mathcal{L} = \{\lambda_1, , ..., \lambda_m\}$
- Classification scheme ("codeframe") $\mathcal{C} = \{c_1, ..., c_n\}$ is the same for all languages
- Scenario common in many multinational organizations / companies and in many multilingual countries
- Three "variants" of this task

1. (Multiple) Mono-lingual Text Classification



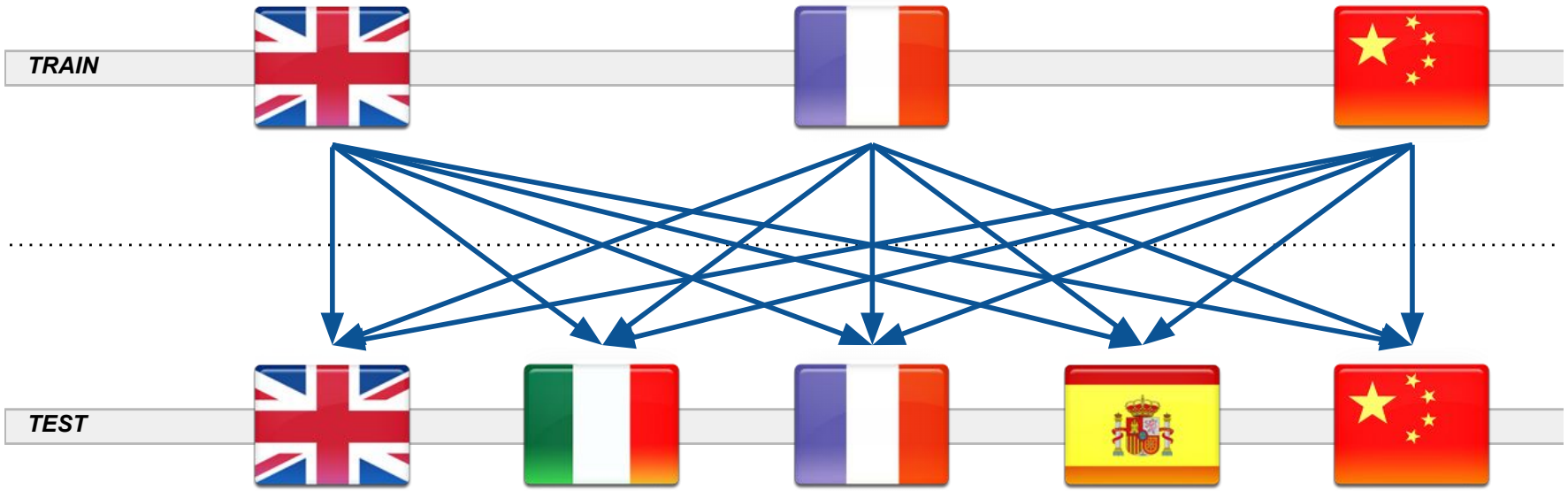
- MLC solved as m independent monolingual classification tasks

2. Poly-lingual Text Classification



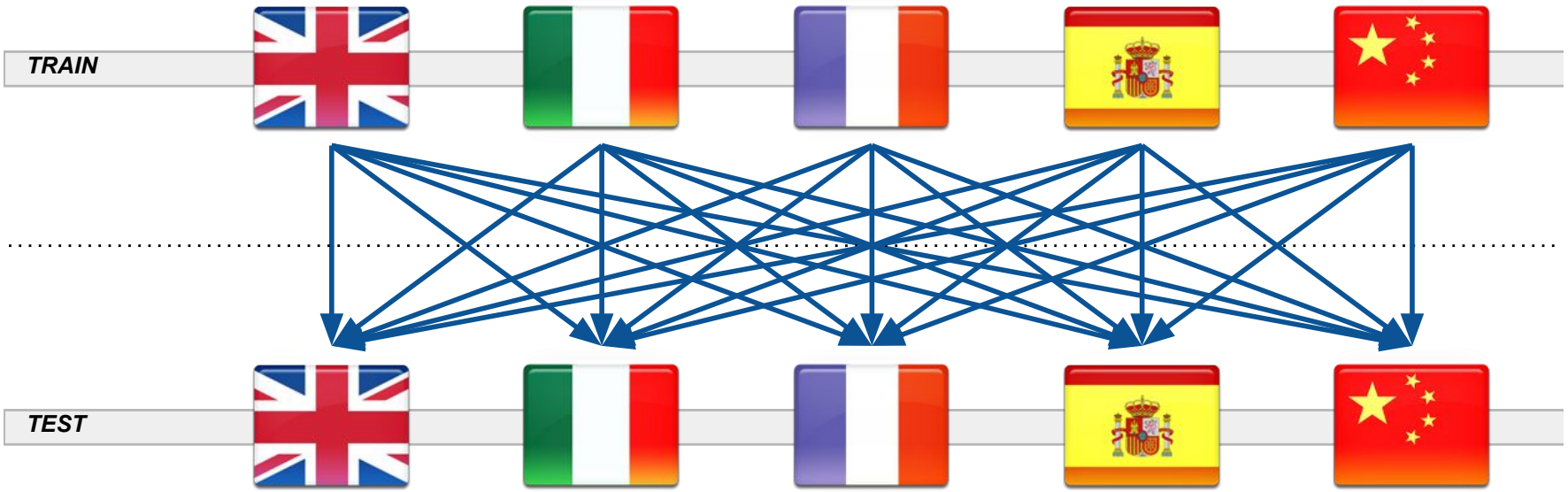
- Attempts to exploit synergies among languages
- Some training examples exist for all languages in $\mathcal{L}$
- Often called the "few-shot" scenario
- $\Rightarrow$ Improve over monolingual classifiers

3. Cross-lingual Text Classification



- Attempts to exploit synergies among languages
- Training examples exist only for the source languages $\mathcal{L}_s \subset \mathcal{L}$ and not for some of the target languages $\mathcal{L}_t \subset \mathcal{L}$
- Often called the "zero-shot" scenario
- $\Rightarrow$ Generate classifiers for languages for which you otherwise could not

Our problem setting



- We will concentrate on polylingual multiclass classification (i.e., $n \geq 2$)
  - single-label PLC (1-of-n), which subsumes the binary case
  - multi-label PLC (any-of-n)

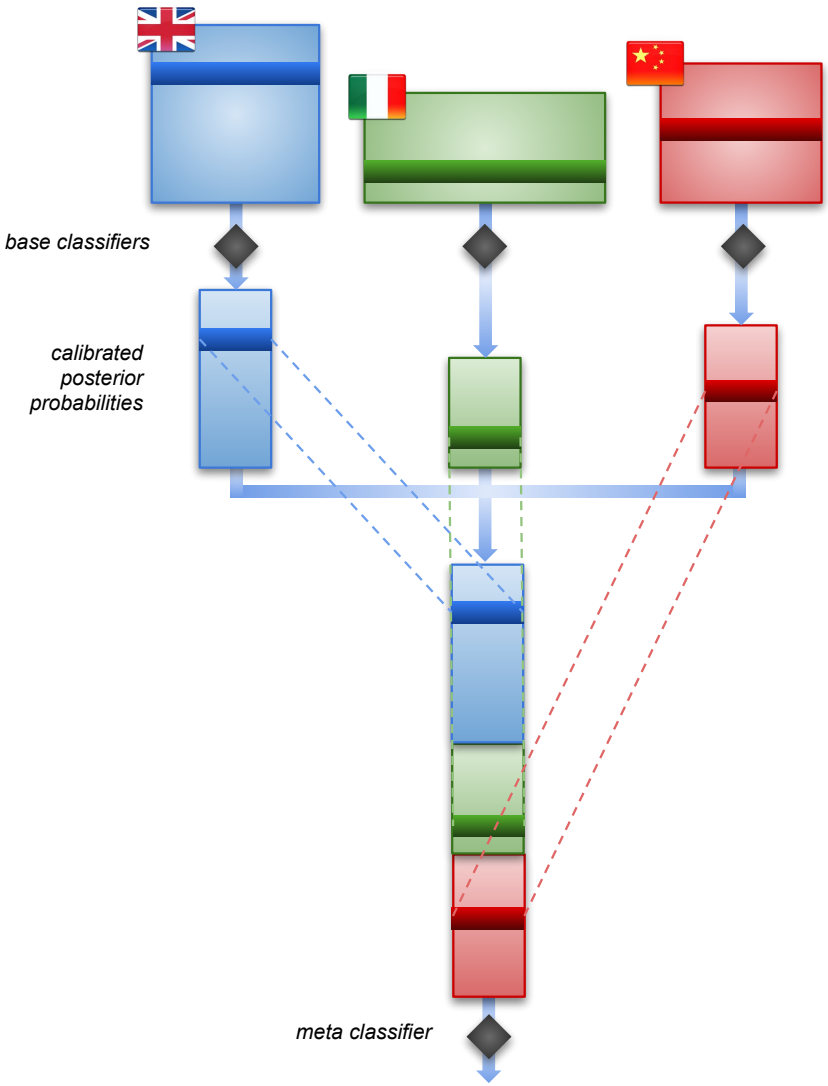
Transfer Learning

- PLC and CLC are instances of (Heterogeneous) Transfer Learning (TL)
- Basic idea of TL: reuse info about a problem in a source domain for solving the same problem in a different target domain
- CLC / PLC : problem = classification in $\mathcal{C}$
  info = training examples
  domain = language
- Useful to address the "training data bottleneck", esp. for under-resourced languages

Transfer Learning

- PLC represents a form of massive TL : all training examples contribute to the classification of all unlabelled examples, irrespectively of language
- How can we achieve that?
- One direction is that of trying to "eliminate the differences between languages"
- Funnelling: a classifier ensemble method for heterogeneous TL

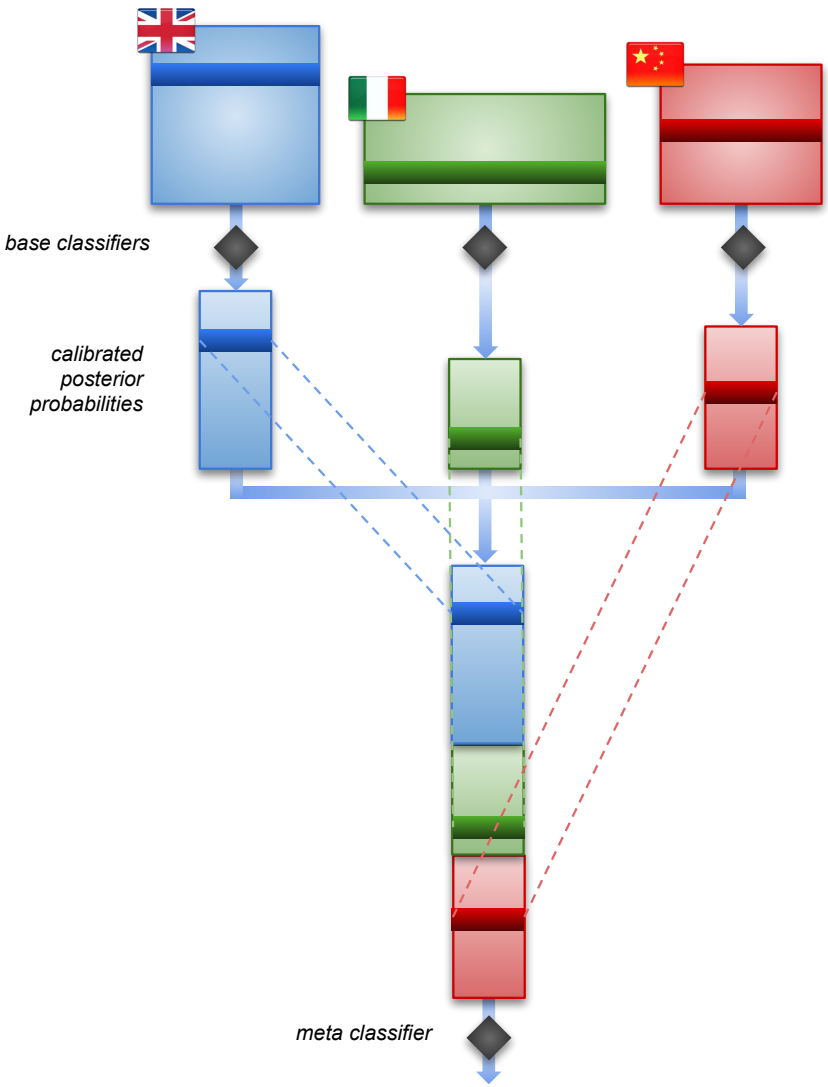
Funnelling: PLC made easy



- Two-level classification architecture
  1. Set of language-dependent base classifiers
  2. Language-independent metaclassifier
- For the metaclassifier, document d represented as vector of n classification scores
- Metaclassifier outputs a vector of n classification scores

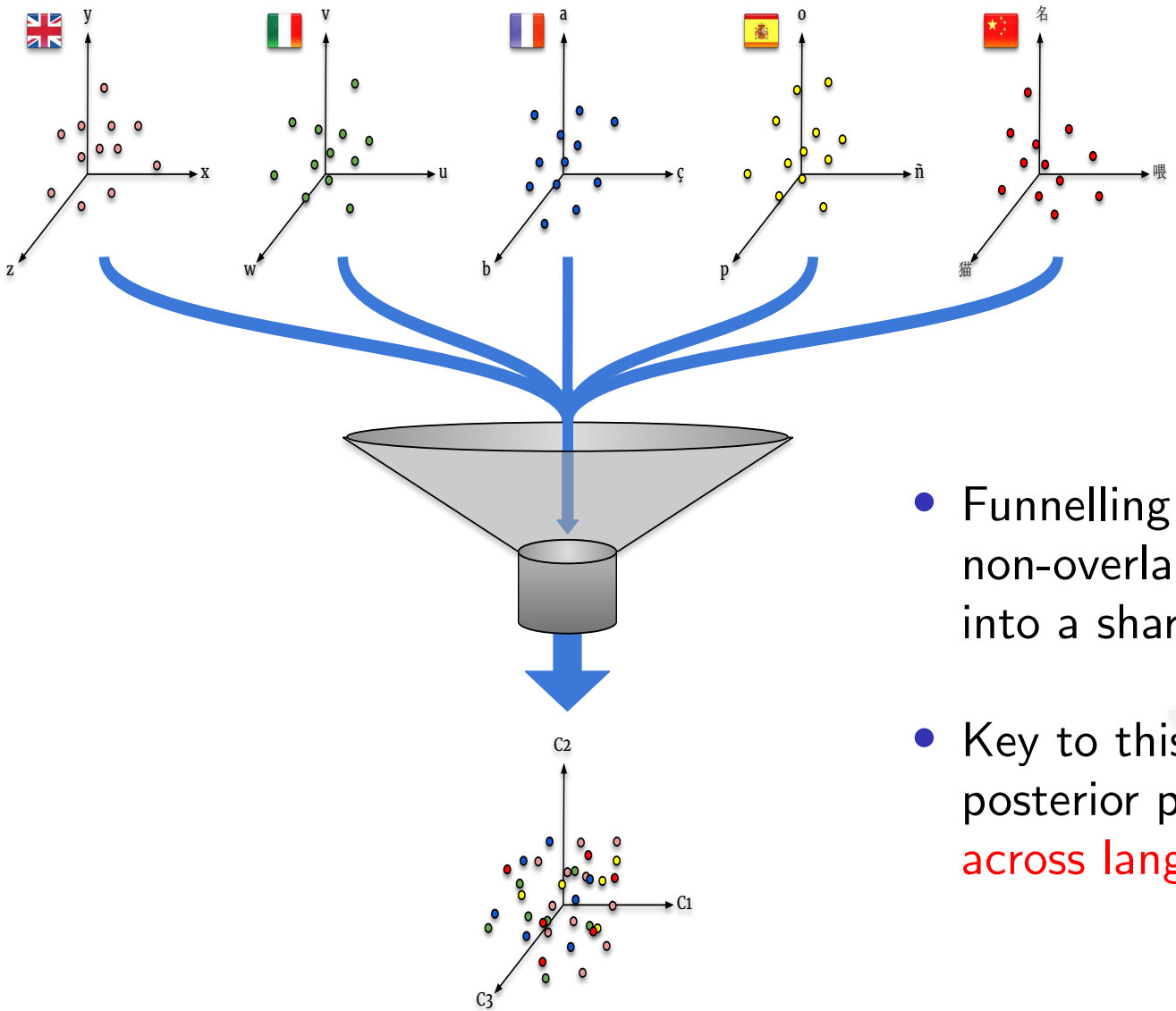
A. Esuli, A. Moreo, F. Sebastiani. Funnelling: A New Ensemble Method for Heterogeneous Transfer Learning and its Application to Cross-Lingual Text Classification. ACM TOIS, 2019.

Funnelling: PLC made easy



- Easy!
- Works for multi-label / single-label / ordinal
- Learner-independent (even allows $\neq$ learners for $\neq$ languages)
- Independent from representation model used in base classifiers (even allows $\neq$ models for $\neq$ languages)
- No requirement that training set should be parallel or comparable
- No requirement for ML dictionaries or MT services

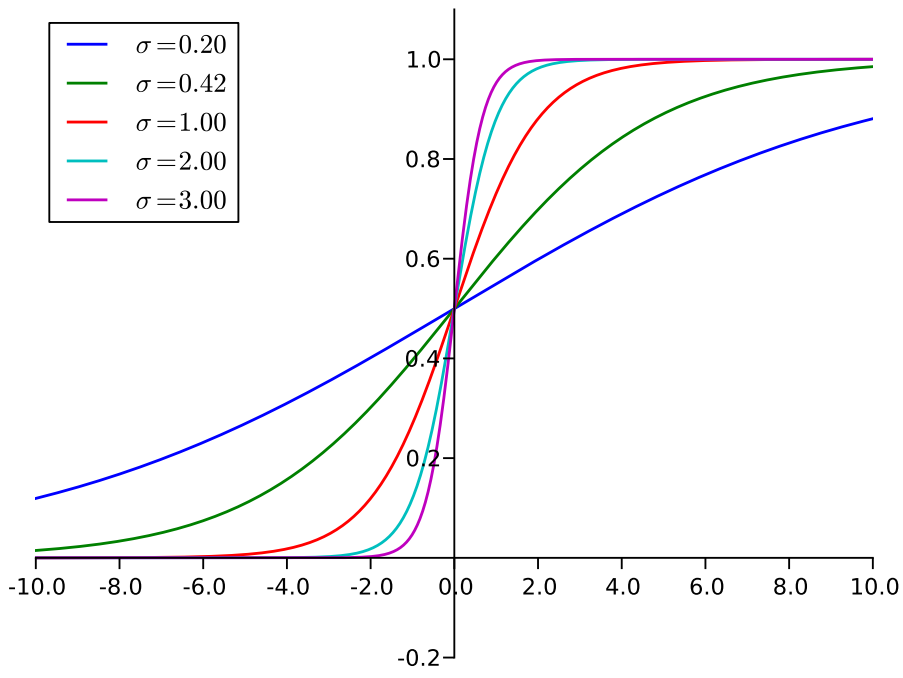
Funnelling: PLC made easy



- Funnelling maps different non-overlapping feature spaces into a shared feature space
- Key to this is the fact that posterior probabilities are aligned across languages

Probability calibration

- Problem: metaclassifier receives, as input, vectors coming from different, incomparable sources
- Solution: make them comparable, by converting classification scores $S(c, d)$ into well calibrated posterior probabilities $\Pr(c|d)$
- Calibration: "90% of items whose $\Pr(c|d)$ is 0.9 should belong to c"
- To be performed independently for each generated classifier



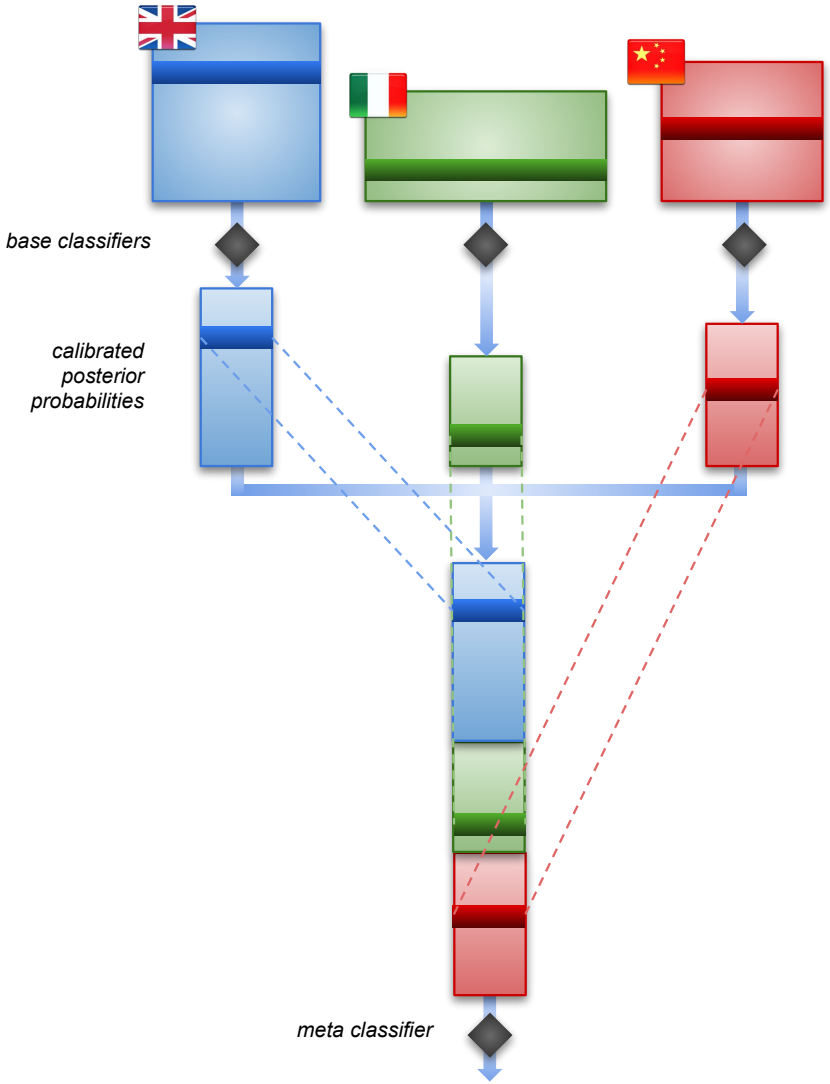
Training a funnelling system

1. Train base classifiers using monolingual training sets
2. Classify training examples
   - via trained classifiers (Fun(TAT))
   - via k-fold cross-validation (Fun(kFCV))
3. Map classification scores into well-calibrated posterior probabilities
4. Use posterior probabilities of training examples for training the metaclassifier

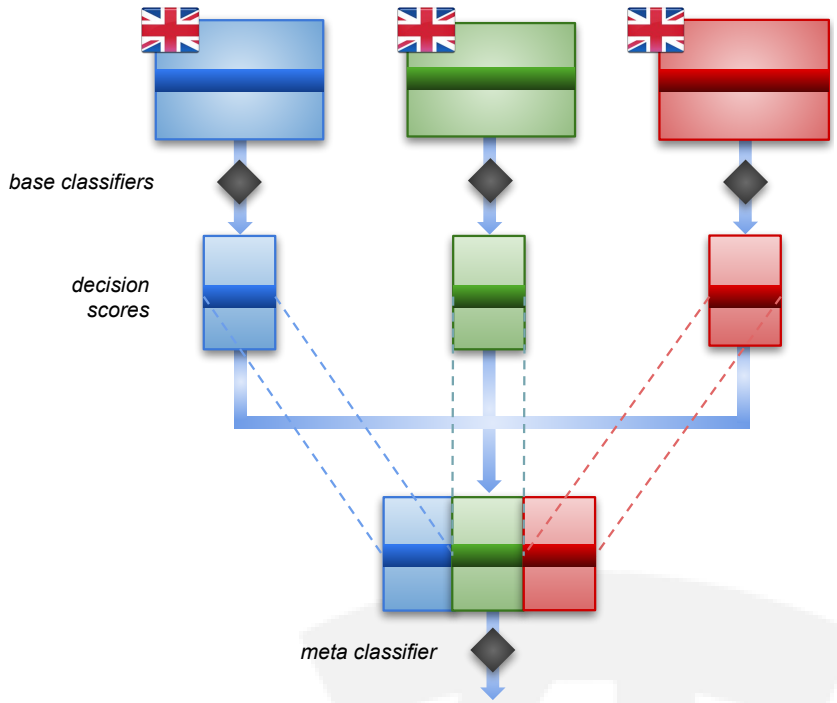
- Fun(TAT): base classifiers generate higher-quality representations for training data than for test data
- Fun(kFCV): base classifiers generate lower-quality representations for training data than for test data
- $\rightarrow$ Choose via experimentation

Funnelling vs. Stacked Generalization

Funnelling



Stacked Generalization

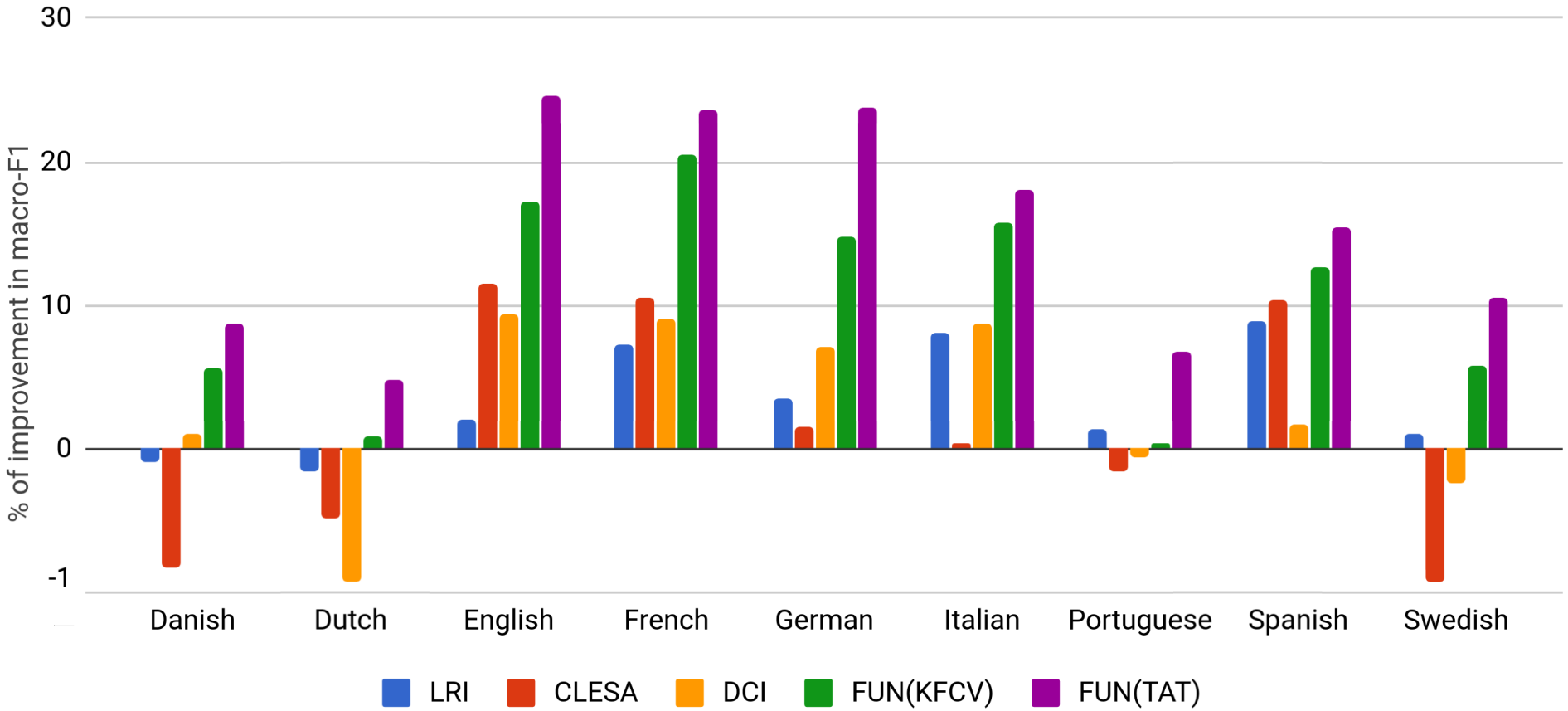


How well does funnelling work?

- Datasets:
  - RCV1/RCV2: comparable corpus, 9 languages, 10 samples $\times$ ((1000 training + 1000 test) per language), 73 classes
  - JRC-Acquis: parallel corpus, 11 languages, 10 samples $\times$ ((1155 training + 4242 test) per language), 300 classes
- Learners:
  - SVMs w/ linear kernel (base classifiers)
  - SVMs w/ RBF kernel (metaclassifier)
- Baselines:
  - Naïve (i.e., multiple monolingual classifiers)
  - Cross-Lingual Explicit Semantic Analysis
  - Distributional Correspondence Indexing
  - Lightweight Random Indexing
- Measures (both in micro- and macro-averaged versions):
  - F_1
  - K ($\approx$ "balanced accuracy")

Some results

- More consistent improvements over naïve baseline



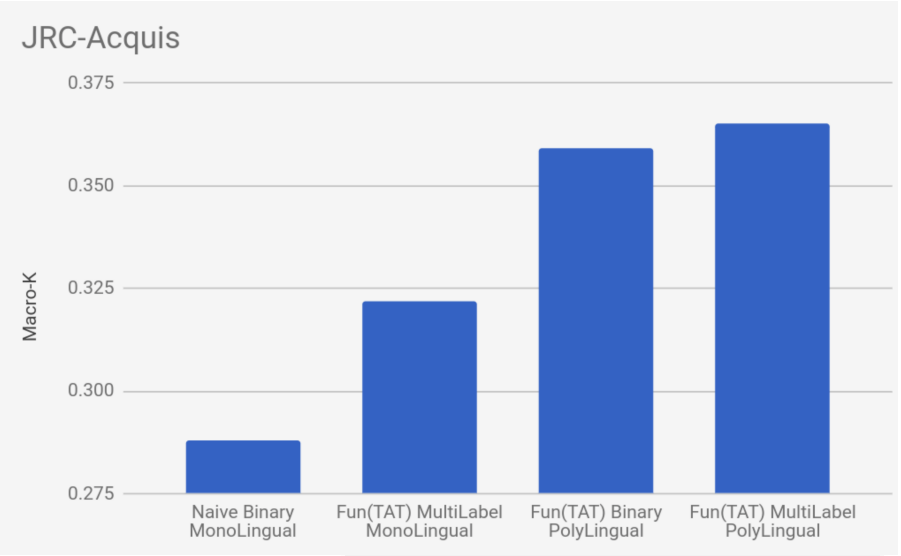
Multi-label PLC results

		NAÏVE	LRI	CLESA	DCI	FUN(KFCV)	FUN(TAT)
F_1^μ	RCV1/RCV2	.776	.771	.714	.770	.801†	**.802**
	JRC-Acquis	.559	**.594**	.557	.510	.581	.587
F_1^M	RCV1/RCV2	.467	.490	.471	.485	.512	**.534**
	JRC-Acquis	.340	**.411**	.379	.317	.356	.399
K^μ	RCV1/RCV2	.690	.696	.659	.696	.731	**.760**
	JRC-Acquis	.429	.476	.453	.382	.457	**.490**
K^M	RCV1/RCV2	.417	.440	.434	.456	.482	**.506**
	JRC-Acquis	.288	.348	.330	.274	.328	**.365**

What does funnelling learn, exactly?

1. ~~The metaclassifier learns to combine scores from different classifiers~~
2. The metaclassifier learns to exploit the stochastic dependencies between classes (the multiclass factor)
3. The metaclassifier learns to classify documents in any language from training documents of any language (the multilanguage factor)

- Which factor contributes most?



Moving further

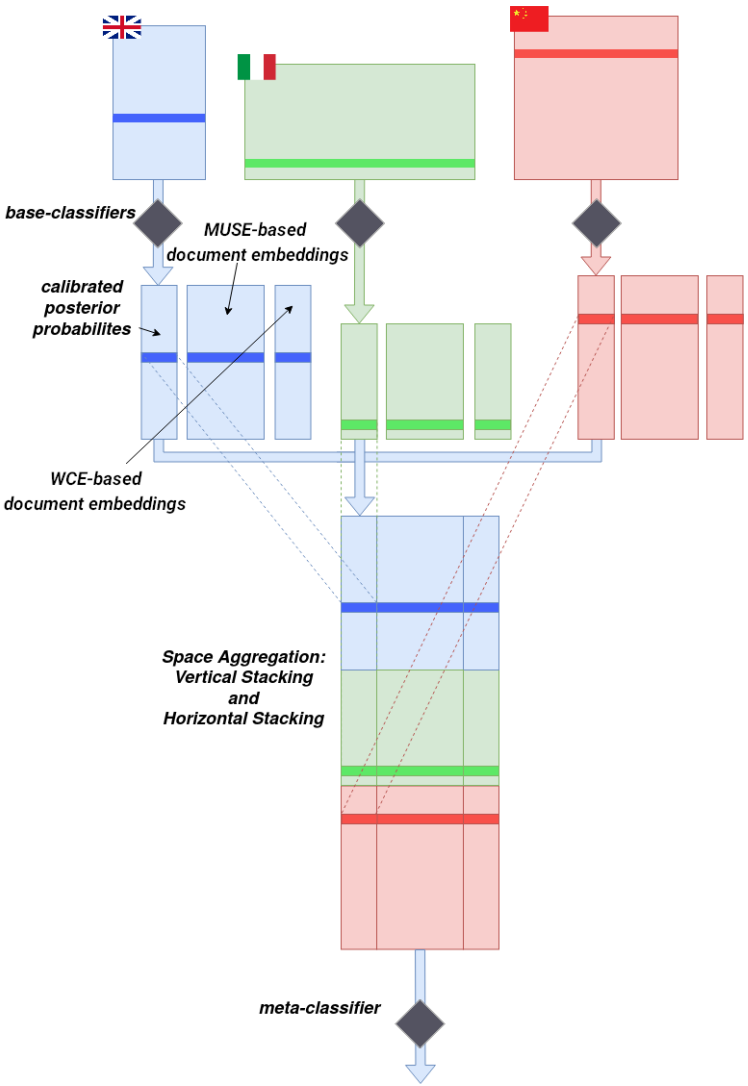
Moving further

- Q: Aside from the vectors of posterior probabilities, can we bring to bear additional knowledge?

Moving further

- Q: Aside from the vectors of posterior probabilities, can we bring to bear additional knowledge?
- A: Yes, as long as this additional knowledge is encoded in vectors aligned across languages
- E.g., class-class correlations in data bring benefit; can we do more?
- We might want to exploit other types of correlations in the data, e.g.,
  - word-class correlations
  - word-word correlations
- Intuition: use different, language-aligned views of the document that encode these correlations

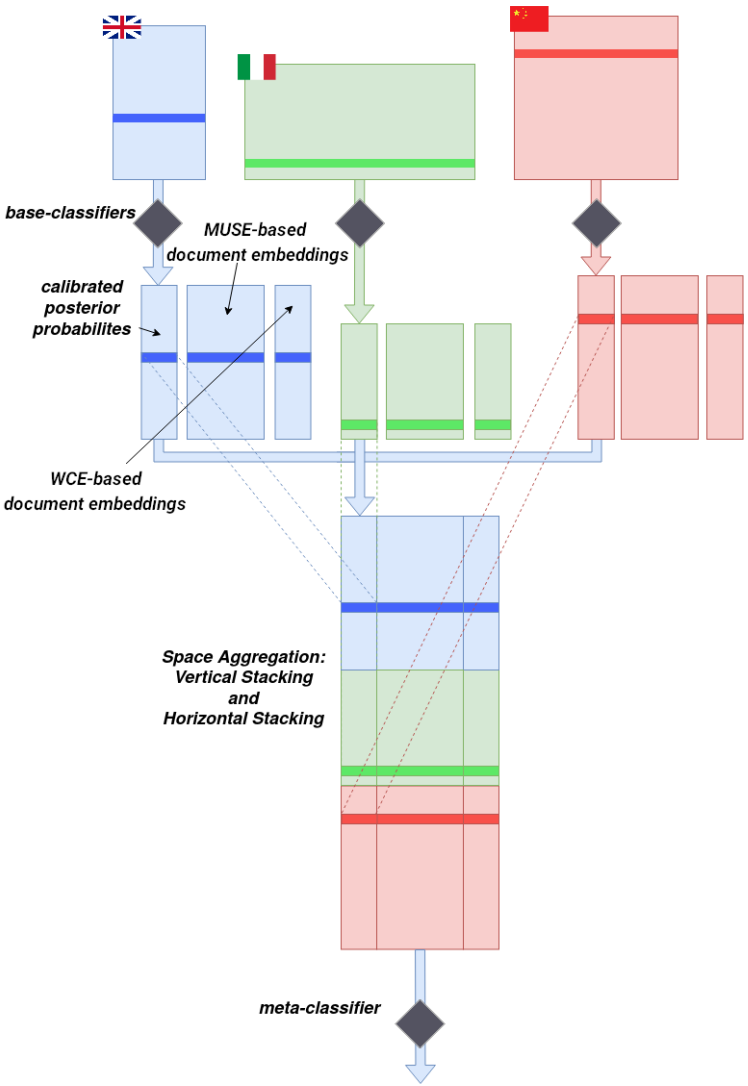
One step further: Generalized Funnelling



- Generalized Funnelling: generalizes 1st-tier classifiers to the notion of view-generating functions (VGFs), i.e., language-dependent functions that generate language-agnostic representations of documents
- We aggregate the representations generated by the different VGFs for the same document into a single vector
- We experiment with three VGFs that return document embeddings aligned across languages

A. Moreo, A. Pedrotti, F. Sebastiani. Funnelling: A New Ensemble Method for Heterogeneous Transfer Learning and its Application to Cross-Lingual Text Classification. ACM SAC 2021.

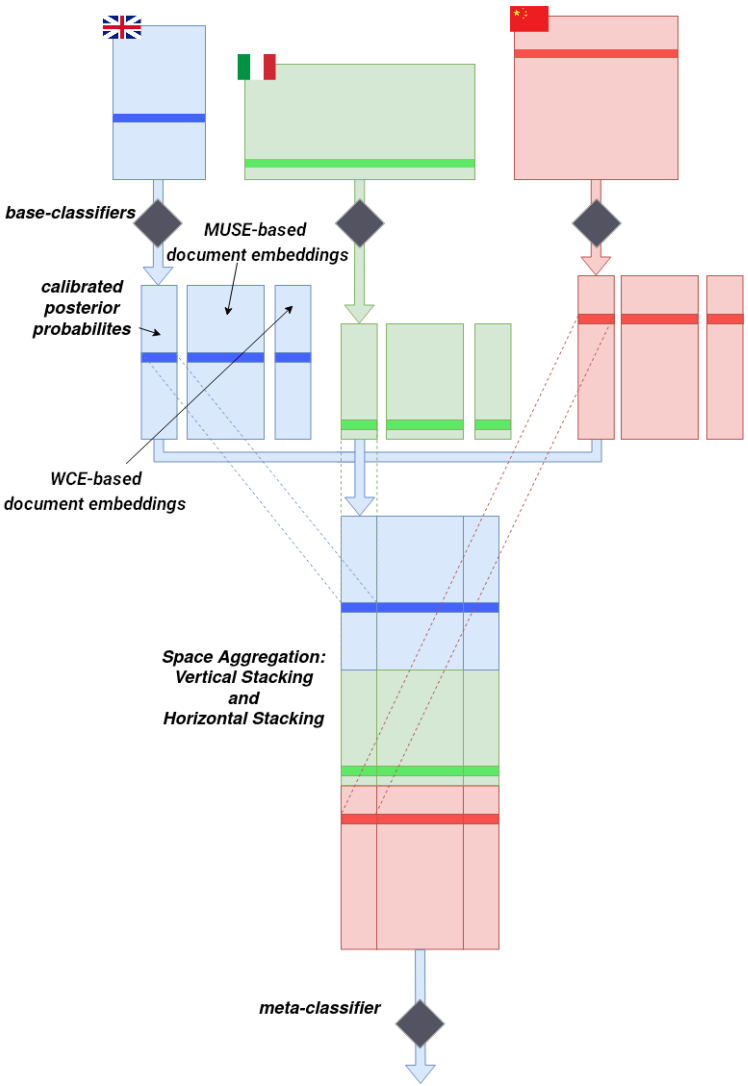
One step further: Generalized Funnelling



- VGF #1: Multilingual Unsupervised/Supervised Embeddings (MUSEs):
  - task-independent word embeddings independently generated for each language and then aligned across languages by an external process
  - capture word-word correlations
  - available for 157 languages from the proposers

A. Conneau, G. Lample, M. Ranzato, L. Denoyer, H. Jégou. 2018. Word translation without parallel data. ICLR 2018.

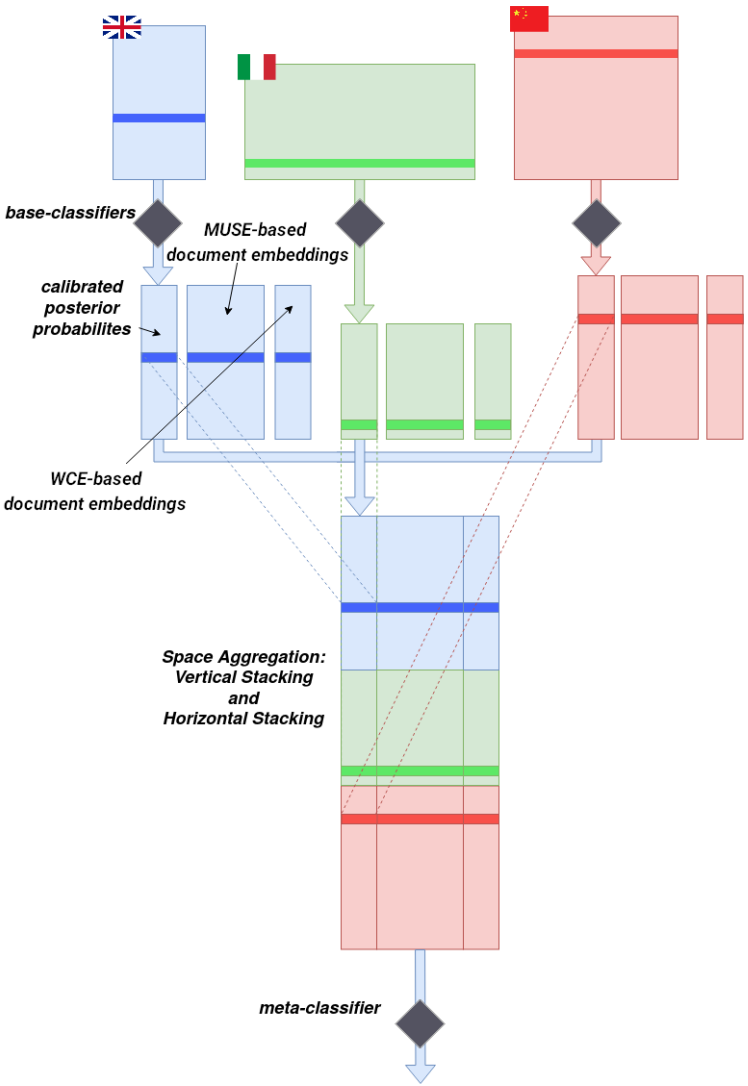
One step further: Generalized Funnelling



- VGF #2: Word-Class Embeddings (WCEs):
  - task-specific word embeddings of size $|\mathcal{C}|$ independently generated for all languages from the language-specific training sets
  - capture word-class correlations
  - natively aligned across languages

A. Moreo, A. Esuli, F. Sebastiani. Word-Class Embeddings for Multiclass Text Classification. Data Mining and Knowledge Discovery, 2021.

One step further: Generalized Funnelling



- VGF #3: multilingual BERT (mBERT):
  - a huge transformer trained on a masked language modelling and next sentence prediction tasks
  - capture word-word correlations in contexts
  - natively aligned across languages

J. Devlin, M.W. Chang, K. Lee, K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of HLT-NAACL 2019

How well does Generalized Funnelling work?

Method	F_1^M	F_1^μ	K^M	K^μ
NAÏVE	.467 ± .083	.776 ± .052	.417 ± .090	.690 ± .074
LRI	.490 ± .077	.771 ± .050	.440 ± .086	.696 ± .069
CLESA	.471 ± .074	.714 ± .061	.434 ± .080	.659 ± .075
DCI	.485 ± .070	.770 ± .052	.456 ± .082	.696 ± .065
FUN	.534 ± .066	.802 ± .041	.506 ±. 073	.760 ± .052
mBERT	.581 ± .014	.817 ± .005	.559 ± .015	.788 ± .008
GFUN–X	.547 ± .065	.798 ± .041	.551 ± .070	.799 ± .046
GFUN–M	.548 ± .066	.769 ± .042	.564 ± .077	.765 ± .048
GFUN–W	.487 ± .062	.743 ± .054	.511 ± .086	.730 ± .058
GFUN–B	.608 ± .064‡	.826 ± .040†	**.603** ± .078	.797 ± .049
GFUN–XMB	**.611** ± .068	**.833** ± .035	.597 ± .077††	**.813** ± .045
GFUN–XWB	.581 ± .062	.821 ± .037	.574 ± .073	.797 ± .046
GFUN–XMW	.558 ± .061	.801 ± .038	.558 ± .072	.788 ± .046
GFUN–WMB	.593 ± .065†	.821 ± .036	.582 ± .079†	.795 ± .048
GFUN–XWMB	.596 ± .064†	.826 ± .035†	.579 ± .075†	.802 ± .046

Conclusion: Where can we go from here?

- Supervised learning tasks different from classification (e.g., multilingual quantification)
- Other classification scenarios (e.g., "multimodal" classification)
- Different codeframes
- Zero-shot classification

Thank you!

Questions?

For any question:

alejandro.moreo@isti.cnr.it

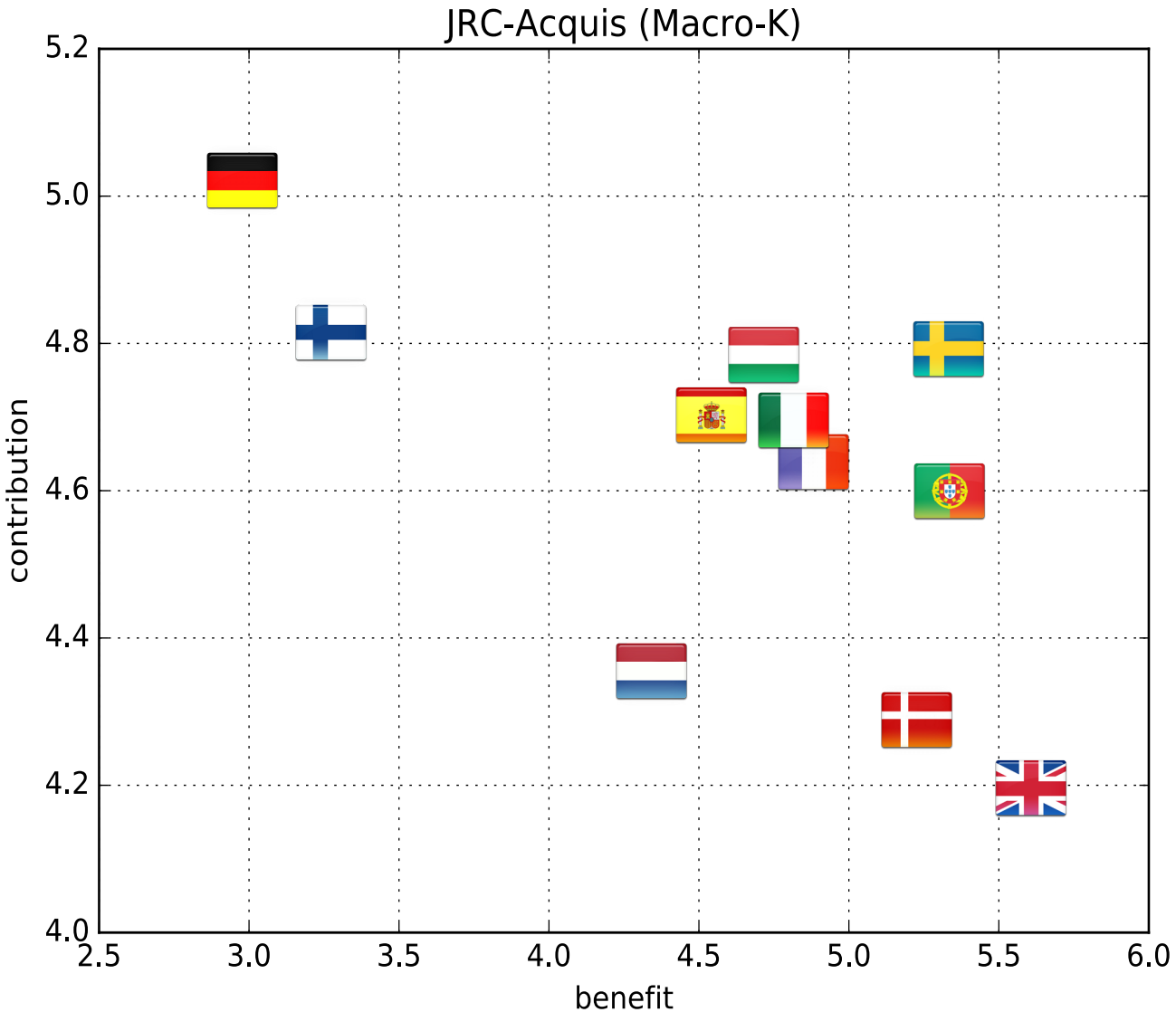
and

fabrizio.sebastiani@isti.cnr.it

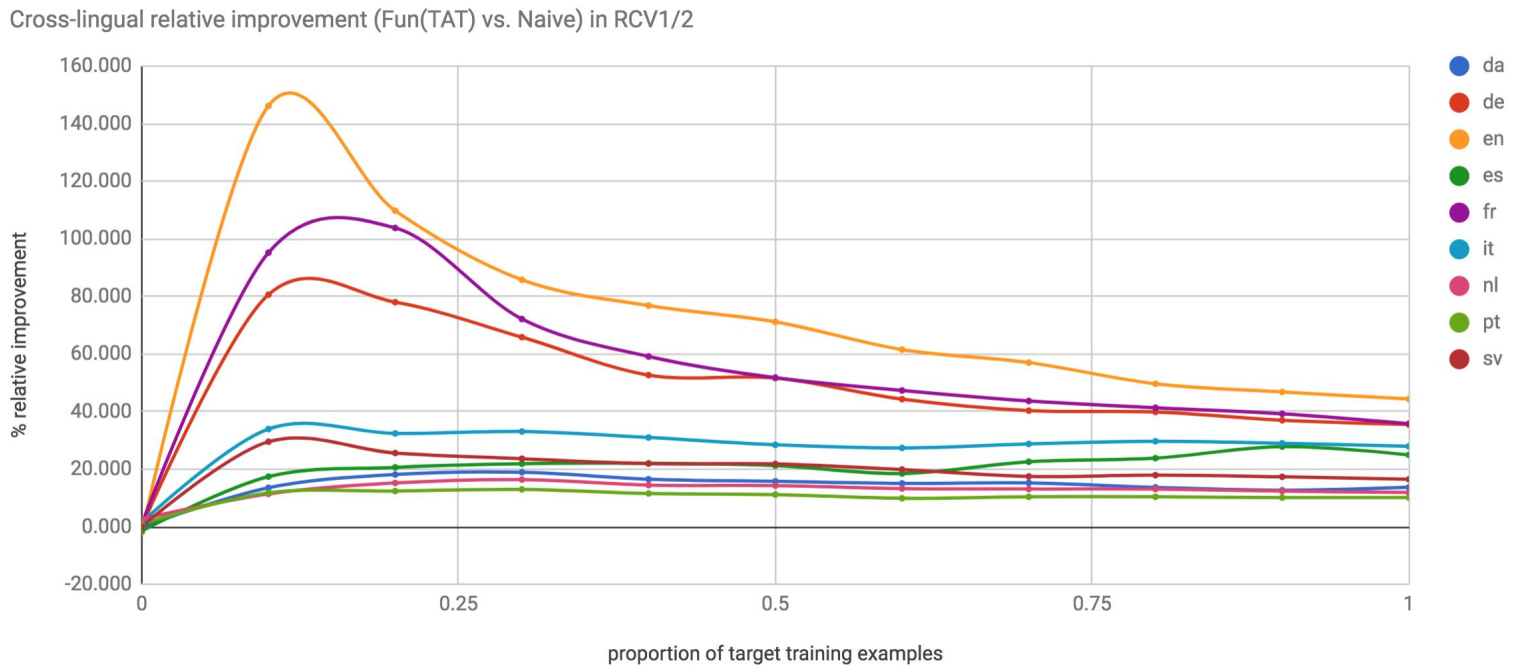
Which factor contributes most?

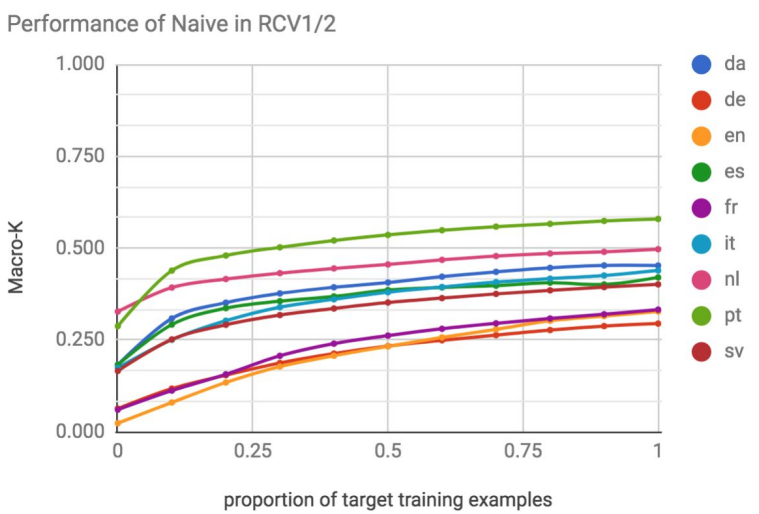
		Naïve Binary MonoLin	Fun(tat) MultiLab MonoLin	Fun(tat) Binary PolyLin	Fun(tat) MultiLab PolyLin
F_1^μ	RCV1/RCV2	.776	.800††	.801††	**.802**
	JRC-Acquis	.559	.573	**.589**	.587††
F_1^M	RCV1/RCV2	.467	.527	.532†	**.534**
	JRC-Acquis	.340	.366	.395††	**.399**
K^μ	RCV1/RCV2	.690	.748	.757	**.760**
	JRC-Acquis	.429	.447	.487††	**.490**
K^M	RCV1/RCV2	.417	.492	.505†	**.506**
	JRC-Acquis	.288	.322	.359	**.365**

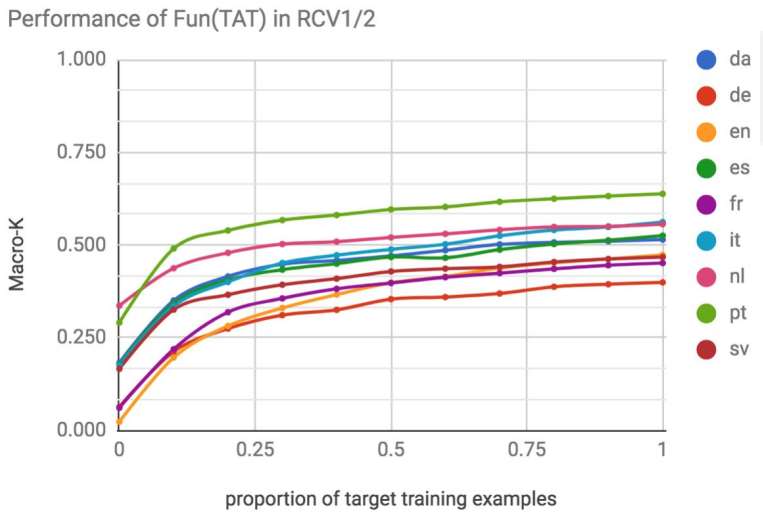
Which languages benefit / contribute most?



How does this contribution evolve?

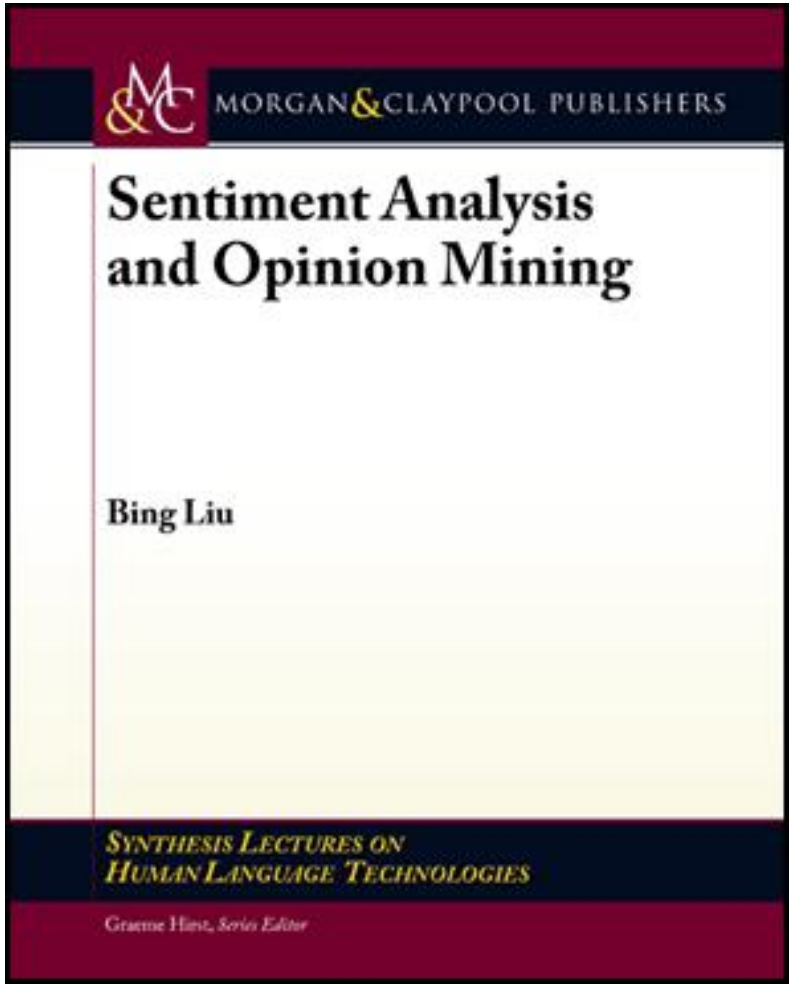






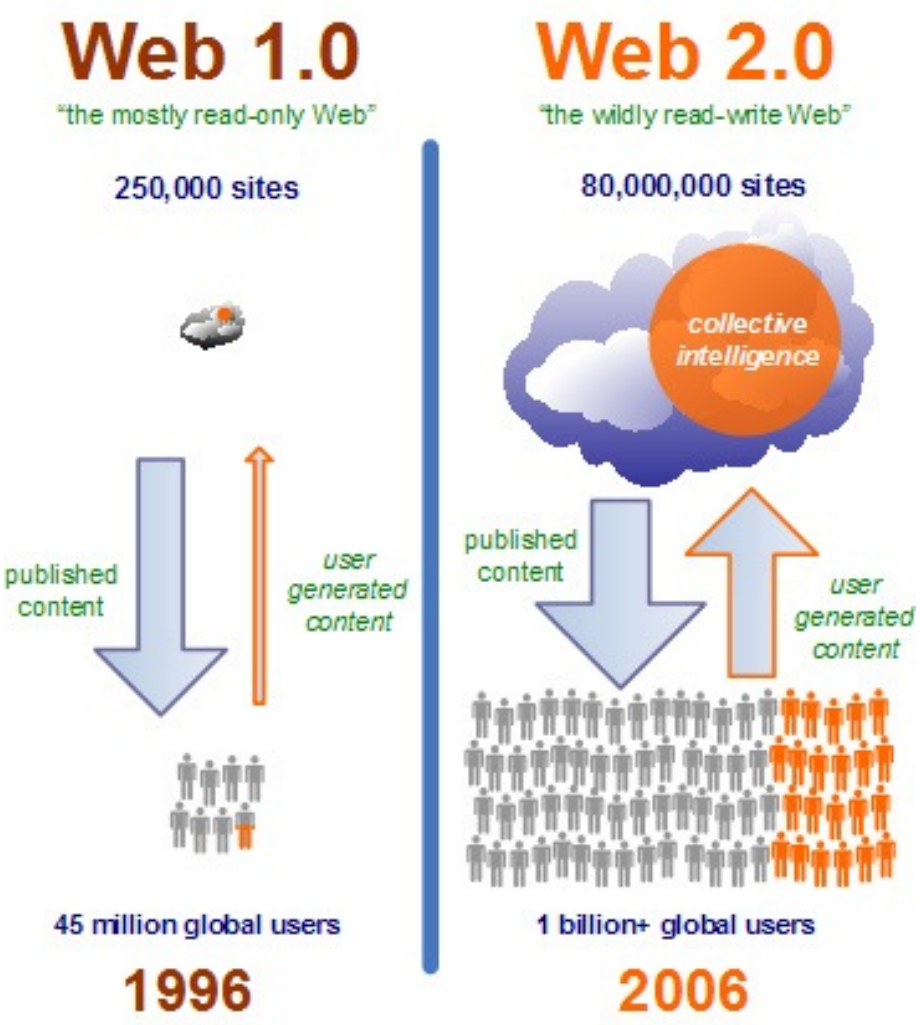
Part IV : Sentiment Analysis

The Task



- Sentiment Analysis and Opinion Mining: a set of tasks concerned with the analysing of texts according to the sentiments / opinions / emotions / judgments (private states, or subjective states) expressed in them
- Originally, term "SA" had a more linguistic slant, while "OM" had a more applicative one
- "SA" and "OM" largely used as synonyms nowadays

Opinion Mining and the Web 2.0 (cont.)



- The 2000's: Web 2.0 is born
- Non-professional users also become authors of content, and this content is often opinion-laden.
- With the growth of UGC, companies understand the value of these data (e.g., product reviews), and generate the demand for technologies capable of mining "sentiment" from them.
- SA becomes the "Holy Grail" of market research, opinion research, and online reputation management.

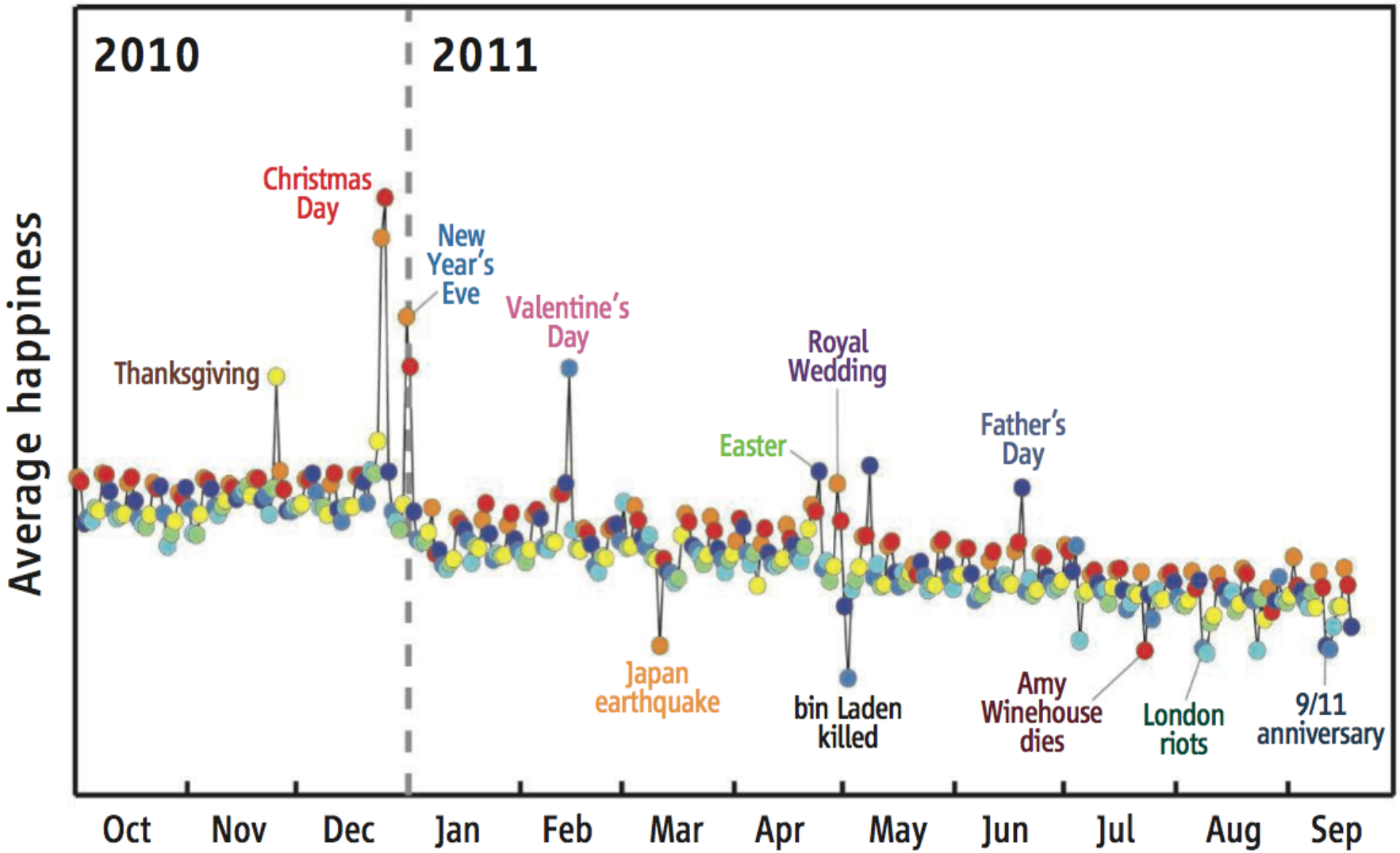
Sentiment Analysis and Opinion Mining

1. The Task
2. Applications of SA and OM
3. The Main Subtasks of SA / OM
4. Advanced Topics

Opinion Research / Market Research via Surveys

- Questionnaires may contain "open" questions
- In many such cases the opinion dimension needs to be analysed, esp. in
  - social sciences surveys
  - political surveys
  - customer satisfaction surveys
- Many such applications are instances of mixed topic / sentiment classification

Computational Social Science



Market Research via Social Media Analysis

HSX.com HOLLYWOOD STOCK EXCHANGE
THE ENTERTAINMENT MARKET

Sign Up | Login
Trade Movies, Stars & More!

MY PORTFOLIO | ON THE EXCHANGE | NEWS & EVENTS | COMMUNITY | EARN H$ | advanced | Search Keyword(s) | GO

The Smiths go to the Weekend Box Office

Predictions for the Weekend of May 31 - June 2, 2013

Title (Distributor)	HSX Market Forecast	HSX "Whisper" Forecast	FilmGo Forecast	HSX Market 4-Week Forecast
After Earth (Sony)	$37.0M	$36.0M	$33.0M	$96.0M
Now You See Me (Summit)	$24.0M	$23.0M	$17.0M	$62.0M

Forecasts as of May 30. Click on the hyperlinked numbers above to see the latest HSX forecasts.

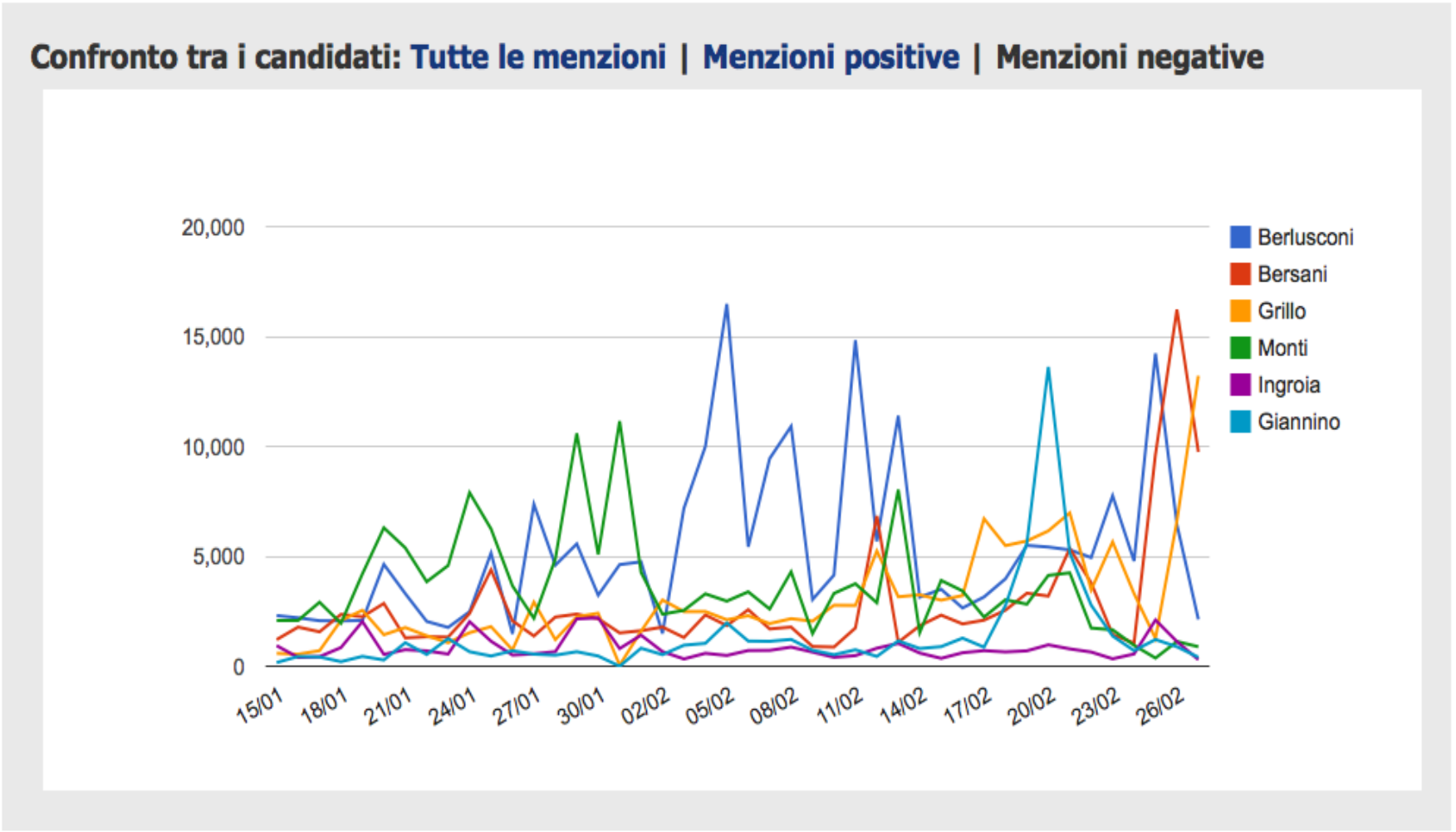
Weekend Predictions for Holdover Films

Predictions for the Weekend of May 31 - June 2, 2013

Title (Distributor)	Week #	Gross to Date	FilmGo Forecast	HSX Market 4-Week Forecast
Fast & Furious 6 (Universal)	2	$130.8M	$38.0M (-61%)	$253.0M
The Hangover Part III (Warner Bros.)	2	$69.4M	$18.8M (-55%)	$115.0M
Epic (Fox)	2	$47.0M	$20.1M (-45%)	$95.0M
Star Trek Into Darkness (Paramount)	3	$162.1M	$20.5M (-45%)	$198.0M
The Great Gatsby (Warner Bros.)	4	$120.8M	$7.3M (-46%)	$134.0M

Forecasts and grosses as of May 30. Click on the hyperlinked numbers above to see the latest HSX forecasts.

Political Science: Predicting Election Results



Online Reputation Detection / Management

Computational Advertising

Sentiment Analysis and Opinion Mining

1. The Task
2. Applications of SA and OM
3. The Main Subtasks of SA / OM
4. Advanced Topics

How Difficult is Sentiment Analysis?

- Sentiment analysis is inherently difficult, because in order to express opinions / emotions / etc. we often use a wide variety of sophisticated expressive means (e.g., metaphor, irony, sarcasm, allegation, understatement, etc.)
  - "At that time, Clint Eastwood had only two facial expressions: with the hat and without it."

    (from an interview with Sergio Leone)
  - "She runs the gamut of emotions from A to B"

    (on Katharine Hepburn in "The Lake", 1934)
  - "If you are reading this because it is your darling fragrance, please wear it at home exclusively, and tape the windows shut."

    (from a 2008 review of parfum "Amarige", Givenchy)
- Sentiment analysis could be characterised as an "NLP-complete" problem

Main Subtasks within SA / OM

- Sentiment Classification: classify a piece of text based on whether it expresses a **Positive** / **Neutral** / **Negative** sentiment
- Sentiment Lexicon Generation: determine whether a word / multiword conveys a **Positive**, **Neutral**, or **Negative** sentiment
- Sentiment Quantification: given a set of texts, estimate the prevalence of different **Positive**, **Neutral**, **Negative** sentiments
- Opinion Extraction (a.k.a. "Fine-Grained SA"): given an opinion-laden sentence, identify the holder of the opinion, its object, its polarity, the strength of this polarity, the type of opinion
- Aspect-Based Sentiment Extraction: given an opinion-laden text about an object, estimate the sentiments conveyed by the text concerning different aspects of the object

Sentiment Classification

- The "queen" of OM tasks
- May be topic-biased or not
  1. Classify items by sentiment; vs.
  2. Find items that express an opinion about the topic, and classify them by their sentiment towards the topic (sometimes called stance classification)
- Binary, ternary, or *n*-ary (ordinal) versions
  - Ternary also involves **Neutral** or **OK-ish** (sometimes confusing the two ...)
  - Ordinal typically uses **1-Star**, **2-Stars**, **3-Stars**, **4-Stars**, **5-Stars** as classes
- At the sentence, paragraph, or document level
  - Classification at the more granular levels used to aid classification at the less granular ones
- May be supervised or unsupervised

Sentiment Classification (cont'd)

- Unsupervised Sentiment Classification (USC) relies on a sentiment lexicon
- The first USC approaches just leveraged the number of occurrences of "positive words" and "negative words" in the text
- Approach later refined in various ways; e.g.,
  - If topic-biased, measure the distance between the sentiment-laden word and a word denoting the topic
  - Bring to bear valence shifters (e.g., particles indicating negated contexts such as `not`, `hardly`, etc.)
  - Bring to bear intensifiers (e.g., `very`, `extremely`) and diminishers (e.g., `fairly`)
  - Bring in syntactic analysis (and other levels of linguistic processing) to determine if sentiment *really* applies to the topic
  - Use WSD in order to better exploit sense-level sentiment lexicons

Sentiment Classification (cont'd)

- Supervised Sentiment Classification (SSC) is just (single-label) text classification with sentiment-related polarities as the classes
- Key fact: bag-of-words (or of-stems, or of-ngrams) does not lead anywhere ...
  - E.g., "`A horrible hotel in a beautiful town!`" vs.
    "`A beautiful hotel in a horrible town!`"
- The same type of linguistic processing used for USC is also needed for SSC, with the goal of generating features for vectorial representations
  $\rightarrow$ "`A` ⟨`Negative`⟩ `hotel in a` ⟨`Positive`⟩ `town!`"
- SSC tends to work better than USC, but requires training data; this has spawned research into
  - Semi-supervised sentiment classification
  - Transfer learning for sentiment classification

Sentiment Lexicon Generation

- The use of a sentiment lexicon is central to both USC and SSC (and to all other OM-related tasks)
- Early sentiment lexicons were small, at the word level, and manually annotated
  - E.g., the General Inquirer
- SLs generated from corpora later became dominant;
  - Some of them are at the word sense level (e.g., SentiWordNet)
  - Some of them are medium-dependent (e.g., SLs for Twitter)
  - Some of them are domain-dependent (e.g., SLs for the financial domain)
  - Many of them are for languages other than English (e.g., SentiWordNet's in other languages)

Sentiment Lexicon Generation (cont'd)

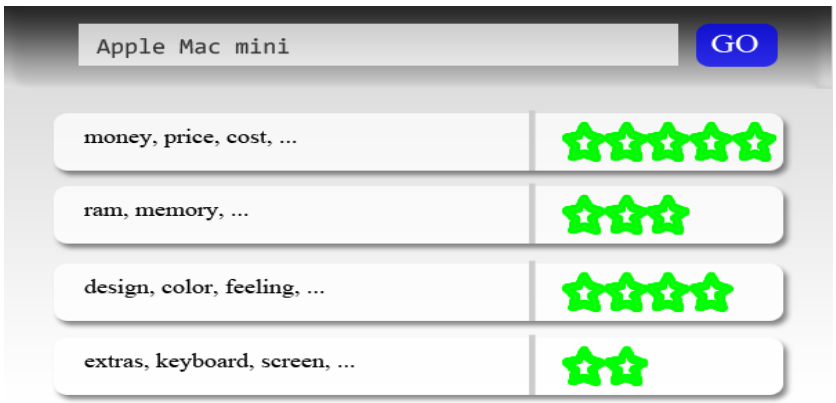
- Several intuitions can be used to generate / extend a SL automatically; e.g.,
  - Conjunctions tend to indicate similar polarity ("`cozy and comfortable`") or opposite polarity ("`small but cozy`") (Hatzivassiloglou and McKeown, 1997)
  - Adjectives highly correlated to adjectives with known polarity tend to have the same polarity (Turney and Littman, 2003)
  - Synonyms (indicated as such in standard thesauri) tend to have the same polarity, while antonyms tend to have opposite polarity (Kim and Hovy, 2004)
  - Sentiment classification of words may be accomplished by classifying their definitions (Esuli and Sebastiani, 2005)
  - Words used in dictionary definitions tend to have the same polarity as the word being defined (Esuli and Sebastiani, 2007)
- The main problem related to SLs is that the polarity of words / word senses is often context-dependent (e.g., `warm blanket` vs. `warm beer`; `low interest rates` vs. `low ROI`)

Opinion Extraction

- Opinion Extraction (a.k.a. "Fine-Grained SA"): given an opinion-laden sentence, identify the holder of the opinion, its object, its polarity, the strength of this polarity, the type of opinion
  - An instance of information extraction, usually carried out via sequence learning (e.g., Conditional Random Fields, HM-SVMs)
  - More difficult than standard IE; certain concepts may be instantiated only implicitly

Aspect-Based Sentiment Extraction

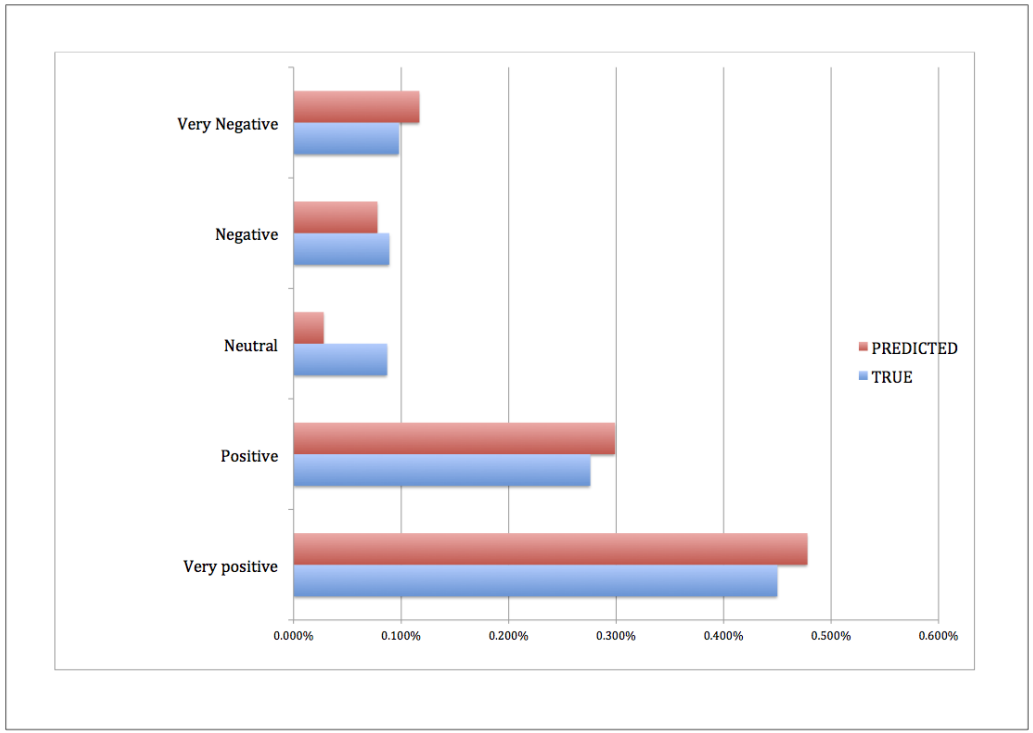
- Aspect-Based Sentiment Extraction: given an opinion-laden text about an object, estimate the sentiments conveyed by the text concerning different aspects of the object
  - Largely driven by need of mining / summarizing product reviews



  - Heavily based on extracting NPs (e.g., `wide viewing angle`) that are highly correlated with the product category (e.g., `Tablet`).
  - Aspects (e.g., `viewing angle`) and sentiments (e.g., `wide`) can be robustly identified via mutual reinforcement

Sentiment Quantification

- In many applications of sentiment classification (e.g., market research, social sciences, political sciences), estimating the relative proportions of **Positive** / **Neutral** / **Negative** documents is the real goal; this is called sentiment quantification[1]
  - E.g., tweets, product reviews



[1] A. Esuli and F. Sebastiani. Sentiment Quantification. *IEEE Intelligent Systems*, 2010.

Sentiment Analysis and Opinion Mining

1. The Task
2. Applications of SA and OM
3. The Main Subtasks of SA / OM
4. Advanced Topics

Advanced Topics in Sentiment Analysis

- Automatic generation of context-sensitive lexicons
- Lexemes as complex objects in sentiment lexicons
- Making sense of sarcasm / irony
- Detecting emotion / sentiment in audio / video using non-verbal features
- Cross-domain / cross-lingual / cross-cultural sentiment analysis

Further Reading

- General:
  - B. Pang, L. Lee: Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2007.
  - B. Liu: *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers, 2012.
  - R. Feldman: Techniques and applications for sentiment analysis. *Communications of the ACM*, 2013.
  - C. Aggarwal: Chapter 13 of *Machine Learning for Text*, Springer, 2018.
- Sentiment analysis in social media
  - S. Kiritchenko, X. Zhu, S. Mohammad: Sentiment Analysis of Short Informal Texts. *Journal of Artificial Intelligence Research* 50, 2014.
  - Martínez-Càmara, E., Martín-Valdivia, M., Urenã López, L., and Montejo Ráez, A. Sentiment analysis in Twitter. *Natural Language Engineering* 20(1), 2014.

Part II :
Learning to Quantify

Learning to Quantify

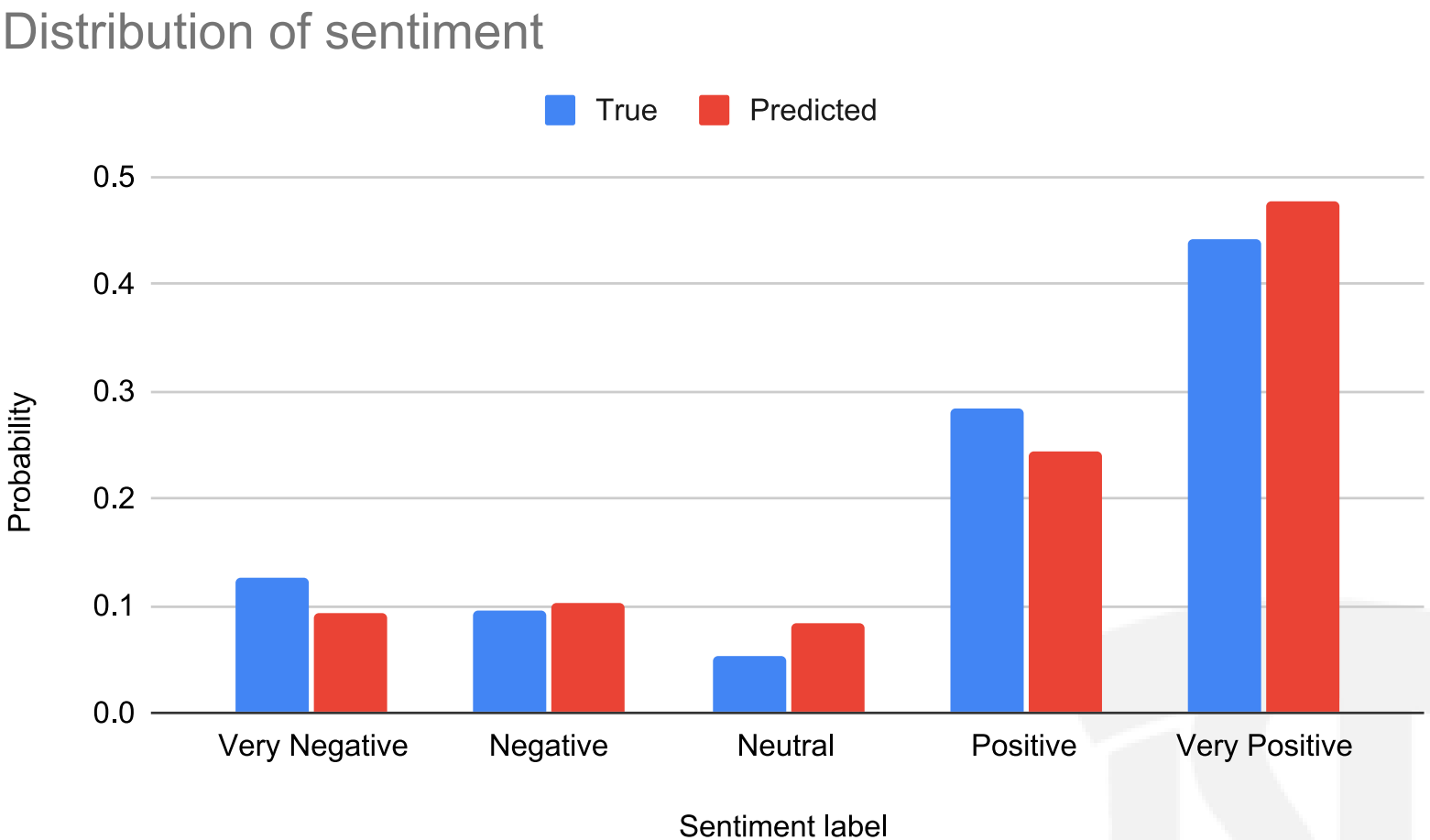
Estimating Class Prevalence via Supervised Learning

Alejandro Moreo and Fabrizio Sebastiani

Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
56124 Pisa, Italy
Email: {firstname.lastname}@isti.cnr.it

ESSIR 2022
Lisbon, Portugal – July 18–22, 2022

What is quantification?



What is quantification? (cont'd)

- In many applications of classification, the real goal is determining the relative frequency (or prevalence) of each class in the unlabelled data; this is called quantification, or supervised prevalence estimation
- Examples:
  - Among the tweets concerning the next midterm elections, what is the percentage of pro-Democrat ones?
  - Among the posts about the Apple Watch 3 posted on forums, what is the percentage of "very negative" ones?
  - How have these percentages evolved over time recently?
- This task has been studied within IR, ML, DM, and has given rise to learning methods and evaluation measures specific to it, and independent from those for classification
- Still a fairly unknown task

P. González, A. Castaño, N. Chawla, and J. del Coz. A review on quantification learning. *ACM Computing Surveys*, 50(5): 74:1–74:40, 2017.

What is quantification? (cont'd)

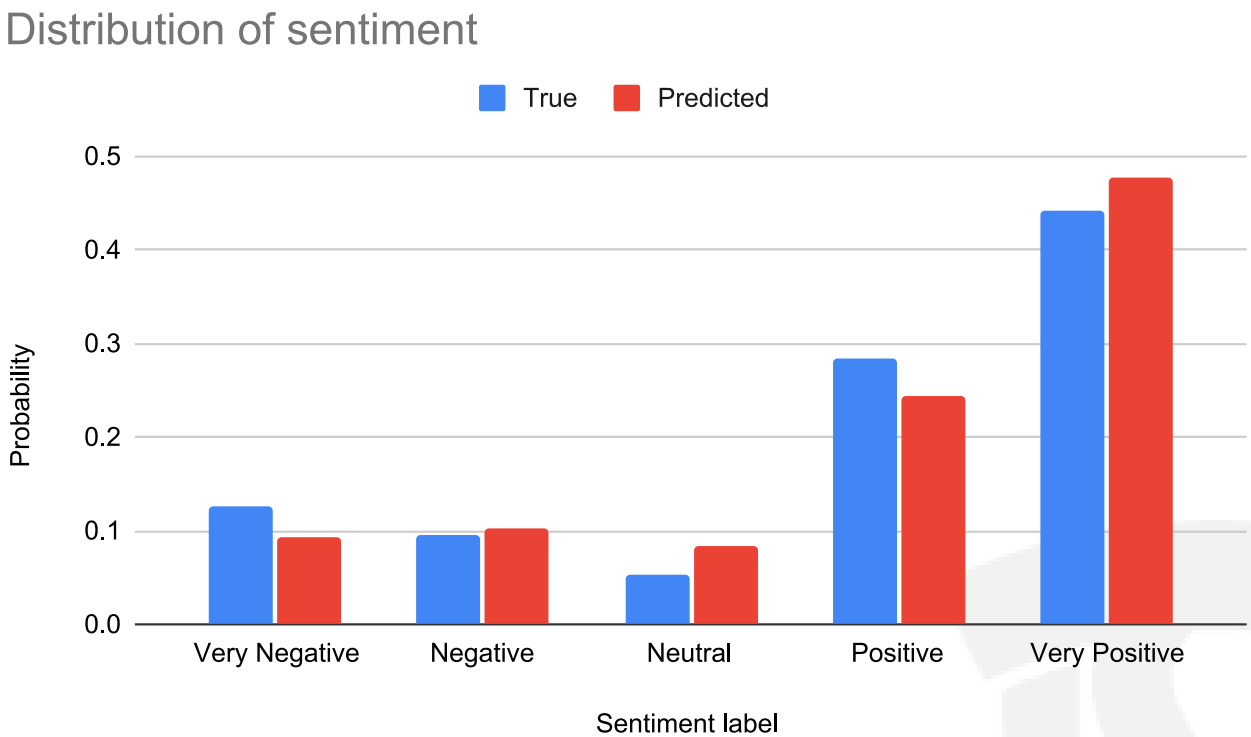
- Quantification goes under different names in different fields
  1. "prevalence estimation" (in statistics and epidemiology)
  2. "class prior estimation" (in machine learning)
  3. "quantification" (in data / text mining)
- "relative frequency" $\equiv$ "prevalence" $\equiv$ "prior probability"
- Slight differences among different fields, e.g.,
  - The task is of independent interest in statistics and data mining, while it is often only ancillary (i.e., functional to generating better classifiers) in machine learning

Structure of this lecture

1. Introduction
2. Applications of quantification in IR, ML, DM, NLP
3. Evaluation measures for quantification
4. Supervised learning methods for quantification
5. Advanced topics
6. Resources and shared tasks
7. Conclusions

What is quantification? (cont'd)

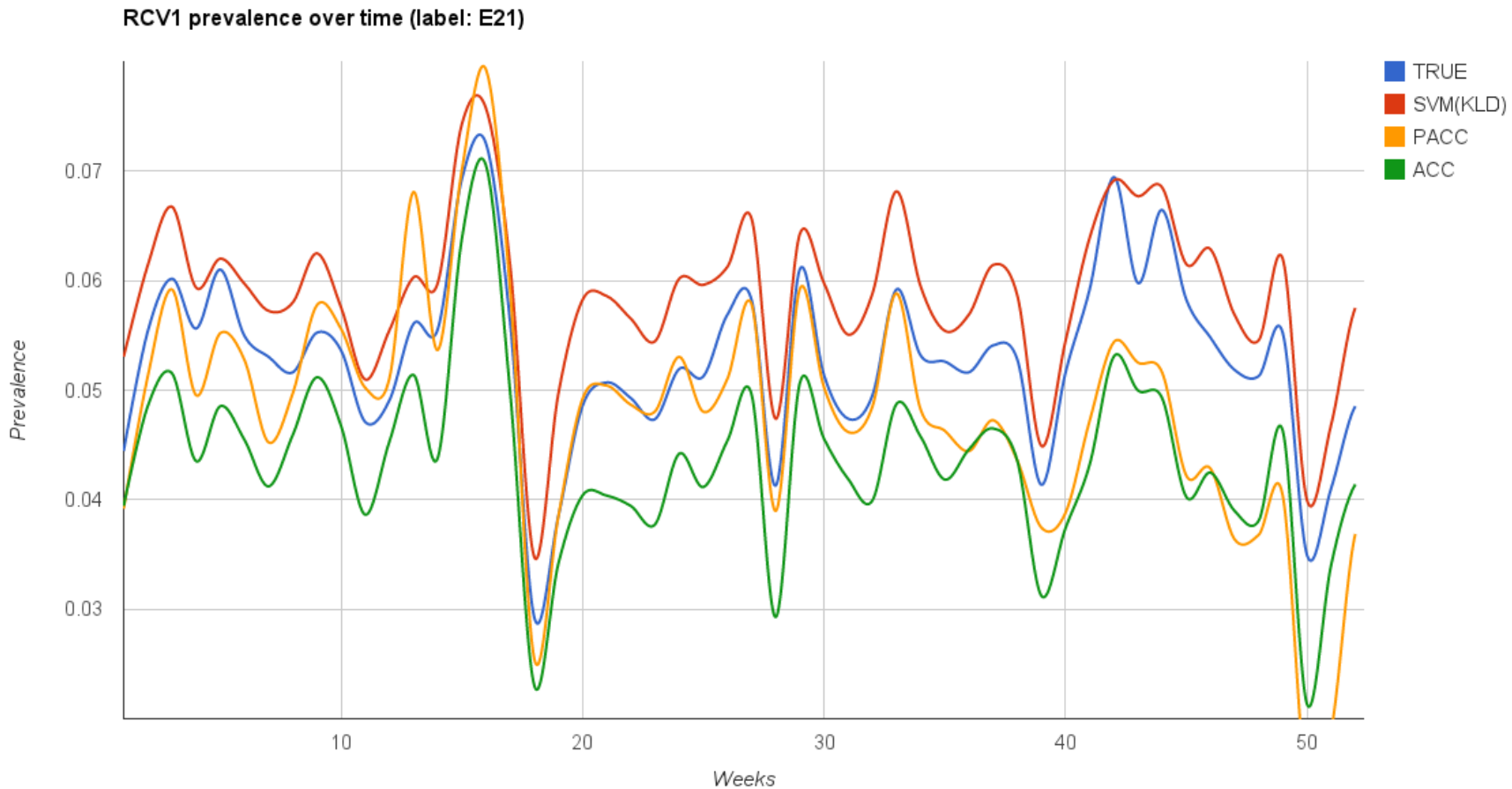
- Quantification may be also defined as the task of approximating a true distribution by a predicted distribution



- As a result, evaluation measures for quantification evaluate how well a predicted distribution approximates the true distribution

Distribution drift

- The need to perform quantification arises because of distribution drift, i.e., the presence of a discrepancy between the class distribution $p_L(y)$ of the training set and the class distribution $p_U(y)$ of the unlabelled set.



Distribution drift (cont'd)

- Distribution drift may derive when
  - the environment is not stationary across time and/or space and/or other variables, and the operating conditions are irreproducible at training time
  - in the presence of sample selection bias, as when the process of labelling training data introduces bias in the training set:
    - explicitly: e.g., by oversampling the minority class
    - implicitly: e.g., if active learning is used
- The presence of distribution drift invalidates the iid assumption, on which standard ML algorithms are instead based; using such algorithms may thus lead to suboptimal quantification accuracy
- Quantification typically assumes the type of drift is prior probability shift, that is, the class prevalence values $\Pr(y)$ can vary, while the conditional probabilities $\Pr(y|\mathbf{x})$ remain stable.

The "paradox of quantification"

- Is "classify and count" the optimal quantification strategy?

The "paradox of quantification"

- Is "classify and count" the optimal quantification strategy? No!
- A perfect classifier is also a perfect "quantifier" (i.e., estimator of class prevalence), but ...
- ... a good classifier is not necessarily a good quantifier (and vice versa):

The "paradox of quantification"

- Is "classify and count" the optimal quantification strategy? No!
- A perfect classifier is also a perfect "quantifier" (i.e., estimator of class prevalence), but ...
- ... a good classifier is not necessarily a good quantifier (and vice versa):

	FP	FN
Classifier h_1	18	20
Classifier h_2	20	20

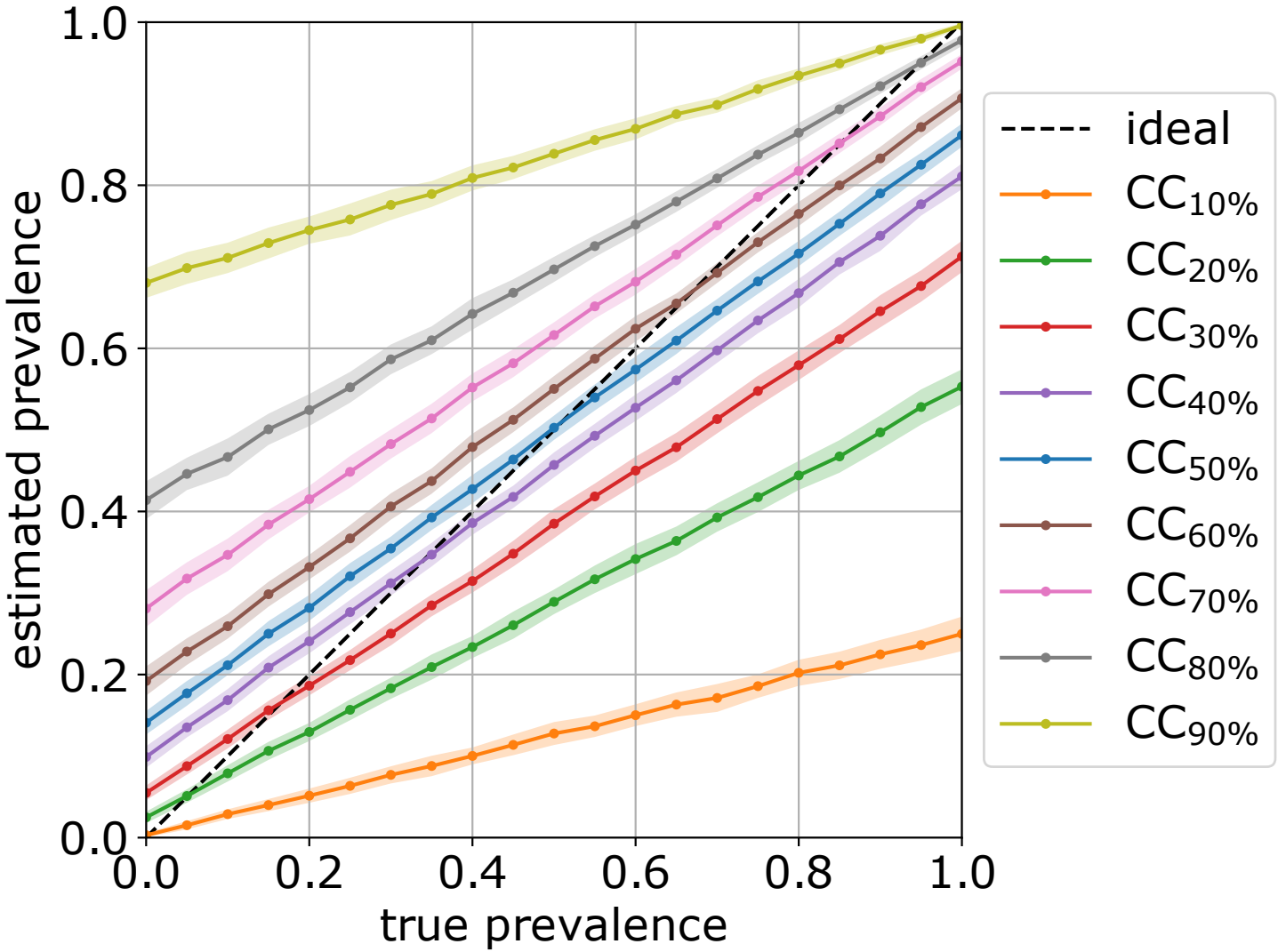
The "paradox of quantification"

- Is "classify and count" the optimal quantification strategy? No!
- A perfect classifier is also a perfect "quantifier" (i.e., estimator of class prevalence), but ...
- ... a good classifier is not necessarily a good quantifier (and vice versa):

	FP	FN
Classifier h_1	18	20
Classifier h_2	20	20

- Paradoxically, we should prefer quantifier h_2 to quantifier h_1, since in quantification a false positive and a false negative cancel each other out
- This means that quantification should be studied as a task in its own right

Bias mitigation



Historical development

- The history of quantification research is highly non-linear (task discovered and re-discovered from within different disciplines)
- 1st stage: interest in the "estimation of class priors" in machine learning
  - Goal: making classifiers robust to the presence of distribution drift and better attuned to the characteristics of the data to which they need to be applied
  - Earliest recorded method is (Vucetic & Obradovic, 2001), most influential one is (Saerens et al., 2002)
- 2nd stage: interest in "quantification" from data mining / text mining
  - Goal: estimating quantities and trends from unlabelled data
  - Earliest recorded work is (Forman, 2005), where the term "quantification" was coined
  - It is the applications from these fields that have provided the impetus behind the most recent wave of research in quantification

G. Forman. Counting positives accurately despite inaccurate classification. ECML 2005.

Structure of this lecture

1. Introduction
2. Applications of quantification in IR, ML, DM, NLP
3. Evaluation measures for quantification
4. Supervised learning methods for quantification
5. Advanced topics
6. Resources and shared tasks
7. Conclusions

Applications of quantification

- A number of fields where classification is used are not interested in individual data, but in data aggregated across spatio-temporal contexts and according to other variables (e.g., gender, age group, religion, job type, ...)

Applications of quantification (cont'd)

- Social sciences: studying indicators concerning society and the relationships among individuals within it

  > *[Others] may be interested in finding the needle in the haystack, but social scientists are more commonly interested in characterizing the haystack.*
  >
  > *(Hopkins and King, 2010)*

  "Computational social science" is the big new paradigm spurred by the availability of big data from social networks

D. Hopkins and G. King. A method of automated nonparametric content analysis for social science. *American Journal of Political Science* 54(1), 2010.

Applications of quantification (cont'd)

- Social sciences: studying indicators concerning society and the relationships among individuals within it

  > *[Others] may be interested in finding the needle in the haystack, but social scientists are more commonly interested in characterizing the haystack.*
  >
  > *(Hopkins and King, 2010)*

  "Computational social science" is the big new paradigm spurred by the availability of big data from social networks
- Political science: e.g., predicting election results / monitoring support for a political party by estimating the prevalence of blog posts / tweets that have a certain stance towards the party

D. Hopkins and G. King. A method of automated nonparametric content analysis for social science. *American Journal of Political Science* 54(1), 2010.

Applications of quantification (cont'd)

- Market Research: estimating the distribution of consumers' attitudes about products, product features, or marketing strategies; e.g.,
  - quantifying customers' attitudes from verbal responses to open-ended questions (Esuli and Sebastiani, 2010)

A. Esuli and F. Sebastiani. Machines that learn how to code open-ended survey data. *International Journal of Market Research* 52(6):775–800, 2010.

Applications of quantification (cont'd)

- Market Research: estimating the distribution of consumers' attitudes about products, product features, or marketing strategies; e.g.,
  - quantifying customers' attitudes from verbal responses to open-ended questions (Esuli and Sebastiani, 2010)
- Epidemiology: tracking the incidence and the spread of diseases; e.g.,
  - estimate pathology prevalence from clinical reports where pathologies are diagnosed
  - estimate the prevalence of different causes of death from "verbal autopsies", i.e., from verbal accounts of symptoms

A. Esuli and F. Sebastiani. Machines that learn how to code open-ended survey data. *International Journal of Market Research* 52(6):775–800, 2010.

Structure of this lecture

1. Introduction
2. Applications of quantification in IR, ML, DM, NLP
3. Evaluation measures for quantification
4. Supervised learning methods for quantification
5. Advanced topics
6. Resources and shared tasks
7. Conclusions

How do we evaluate quantification methods?

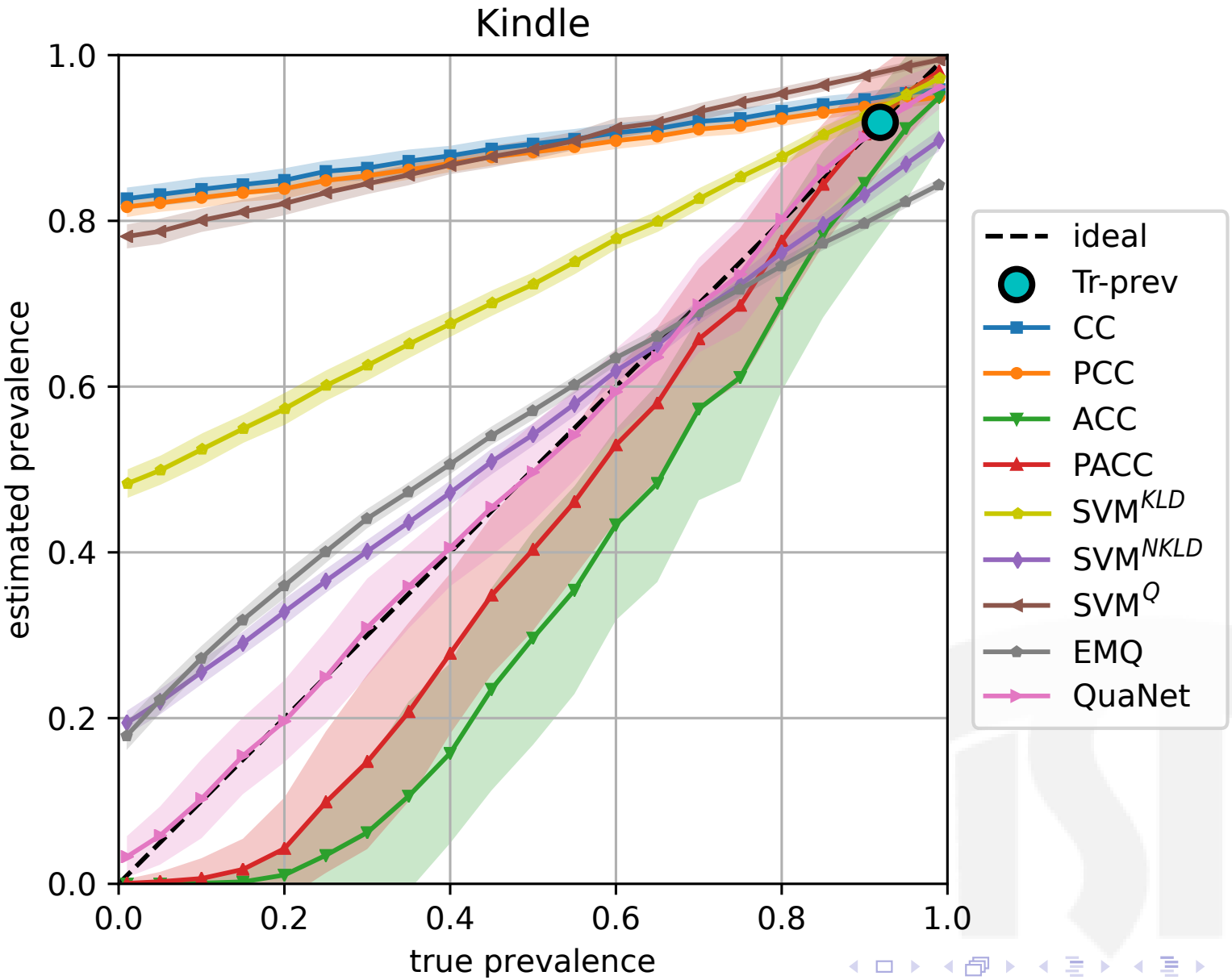
- Evaluating quantification means measuring how well a predicted probabilistic distribution $\hat{p}(y)$ fits a true distribution $p(y)$
- The goodness of fit (or lack thereof) between two categorical distributions can be computed via divergence functions
- Divergences frequently used for evaluating (SLMC) quantification are
  - $\mathrm{AE}(p, \hat{p}) = \frac{1}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} |\hat{p}(y) - p(y)|$ (Absolute Error)
  - $\mathrm{RAE}(p, \hat{p}) = \frac{1}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} \frac{|\hat{p}(y) - p(y)|}{p(y)}$ (Relative Absolute Error)
  - $\mathrm{KLD}(p, \hat{p}) = \sum_{y \in \mathcal{Y}} p(y) \log \frac{p(y)}{\hat{p}(y)}$ (Kullback-Leibler Divergence)

F. Sebastiani. Evaluation measures for quantification: An axiomatic approach, *Information Retrieval Journal*, 2019.

Experimental protocols for evaluating quantification

- Standard classification datasets may be used for evaluating quantification
- Two different experimental protocols used in the literature
  - the artificial-prevalence protocol: take a standard dataset split into L and U, and conduct repeated experiments in which the $p_U(y_j)$'s are artificially varied via undersampling or oversampling
    - **Pros**: class prevalence / drift may be varied at will
    - **Cons**: non-realistic experimental settings may result
      not obvious how to implement it in the SLMC case
  - the natural-prevalence protocol: pick one or more standard datasets that represent a wide array of class prevalences and drifts
    - **Pros**: experimental settings tested are realistic
    - **Cons**: class prevalence and drift may not be varied at will
      hard and expensive to obtain

The artificial-prevalence protocol: An example



Structure of this lecture

1. Introduction
2. Applications of quantification in IR, ML, DM, NLP
3. Evaluation measures for quantification
4. Supervised learning methods for quantification
5. Advanced topics
6. Resources and shared tasks
7. Conclusions

Quantification methods

- Quantification methods belong to three classes
  - 1. Aggregative: require the classification of individual items as a basic intermediate step
  - 2. Non-aggregative: quantification is performed without classifying the individual items
  - 3. Ensembles: rely on an ensemble of (aggregative or non-aggregative) quantifiers
- Aggregative methods may be further subdivided into
  - 1a. Methods using general-purpose learners (i.e., learners originally devised for classification)
  - 1b. Methods using special-purpose learners (i.e., learners especially devised for quantification)

Quantification methods: CC

- Classify and Count (CC) consists of
  1. generating a classifier h from L
  2. classifying the items in U using h
  3. estimating $p_U(y_j)$ by counting the items predicted to be in y_j, i.e.,

$$\hat{p}_U^{CC}(y_j) = \frac{1}{|U|}|\{\mathbf{x} \in U : h(\mathbf{x}) = y_j\}|$$

- But a good classifier is not necessarily a good quantifier ...
- CC suffers from the problem that "standard" classifiers are usually tuned to minimize $(\mathrm{FP} + \mathrm{FN})$ or a proxy of it, but not $|\,\mathrm{FP} - \mathrm{FN}\,|$

Quantification methods: PCC

- Probabilistic Classify and Count (PCC – Bella et al., 2010, but first evoked in Lewis, 1995) estimates p_U by simply counting the expected fraction of items predicted to be in the class, i.e.,

$$\hat{p}_U^{PCC}(y_j) = E_U[y_j] = \frac{1}{|U|} \sum_{\mathbf{x} \in U} p(y_j|\mathbf{x})$$

- The rationale is that posterior probabilities contain richer information than binary decisions, which are obtained from posterior probabilities by thresholding.

A. Bella, C. Ferri, J. Hernańdez-Orallo, M.J. Ramírez-Quintana. Quantification via probability estimators. ICDM 2010, pp. 737–742.

D.D. Lewis. Evaluating and optimizing autonomous text classification systems. SIGIR 1995.

Quantification methods: PCC and Calibration

- PCC requires the classifier to return calibrated posterior probabilities (e.g., LR), i.e., probabilities $p(y_j|\mathbf{x})$ such that

$$\lim_{|S|\to\infty} \frac{|\{\mathbf{x} \in y_j | p(y_j|\mathbf{x}) = \alpha\}|}{|\{\mathbf{x} \in S | p(y_j|\mathbf{x}) = \alpha\}|} = \alpha \tag{1}$$

- E.g., 82% of the instances $\mathbf{x}$ for which $p(y_j|\mathbf{x}) = 0.82$, belong to y_j
- Confidence scores $s_j(\mathbf{x})$ that are not probabilities (e.g., SVMs) or are non-calibrated probabilities (e.g., NB) must be converted into calibrated posterior probabilities, e.g., by applying a sigmoidal (e.g., logistic) function

$$p(y_j|\mathbf{x}) = \frac{1}{1 + e^{\sigma s_j(\mathbf{x}) + \theta}}$$

- Calibration consists in tuning σ and θ so that (1) holds

Quantification methods: ACC

- Adjusted Classify and Count (ACC, a.k.a. the Confusion Matrix Method – Forman, 2005) is based on the observation that, after we have classified the unlabelled documents in U, for all $y_i \in \mathcal{Y}$ it holds that

$$\Pr(h(\mathbf{x}) = y_i) = \sum_{y_j \in \mathcal{Y}} \Pr(h(\mathbf{x}) = y_i | y_j) \cdot \Pr(y_j)$$

G. Forman. Counting positives accurately despite inaccurate classification. ECML 2005.

Quantification methods: ACC

- Adjusted Classify and Count (ACC, a.k.a. the Confusion Matrix Method – Forman, 2005) is based on the observation that, after we have classified the unlabelled documents in U, for all $y_i \in \mathcal{Y}$ it holds that

$$\Pr(h(\mathbf{x}) = y_i) = \sum_{y_j \in \mathcal{Y}} \Pr(h(\mathbf{x}) = y_i | y_j) \cdot \Pr(y_j)$$

- The $\Pr(h(\mathbf{x}) = y_i)$'s are observed.

G. Forman. Counting positives accurately despite inaccurate classification. ECML 2005.

Quantification methods: ACC

- Adjusted Classify and Count (ACC, a.k.a. the Confusion Matrix Method – Forman, 2005) is based on the observation that, after we have classified the unlabelled documents in U, for all $y_i \in \mathcal{Y}$ it holds that

$$\Pr(h(\mathbf{x}) = y_i) = \sum_{y_j \in \mathcal{Y}} \Pr(h(\mathbf{x}) = y_i | y_j) \cdot \Pr(y_j)$$

- The $\Pr(h(\mathbf{x}) = y_i)$'s are observed.
- The $\Pr(h(\mathbf{x}) = y_i | y_j)$'s can be estimated on L via k-fold cross-validation (these latter represent the system's "class-conditional bias").

G. Forman. Counting positives accurately despite inaccurate classification. ECML 2005.

Quantification methods: ACC

- Adjusted Classify and Count (ACC, a.k.a. the Confusion Matrix Method – Forman, 2005) is based on the observation that, after we have classified the unlabelled documents in U, for all $y_i \in \mathcal{Y}$ it holds that

$$\Pr(h(\mathbf{x}) = y_i) = \sum_{y_j \in \mathcal{Y}} \Pr(h(\mathbf{x}) = y_i | y_j) \cdot \Pr(y_j)$$

- The $\Pr(h(\mathbf{x}) = y_i)$'s are observed.
- The $\Pr(h(\mathbf{x}) = y_i | y_j)$'s can be estimated on L via k-fold cross-validation (these latter represent the system's "class-conditional bias").
- This results in a system of $|\mathcal{Y}|$ linear equations (one for each y_j) with $|\mathcal{Y}|$ unknowns (the $\Pr(y_j)$'s).
- ACC consists of solving this system, i.e., of correcting the class prevalence estimates $p_U^{CC}(y_i)$ obtained by CC according to the estimated system's bias.

G. Forman. Counting positives accurately despite inaccurate classification. ECML 2005.

Quantification methods: ACC (binary case)

- In the binary case this comes down to

$$\begin{aligned} p_U^{CC}(y_1) &= \Pr(h(\mathbf{x}) = y_1|y_1)\Pr(y_1) + \Pr(h(\mathbf{x}) = y_1|y_2)\Pr(y_2) \\ &= \mathrm{TPR}\, p_U(y_1) + \mathrm{FPR}\, p_U(y_2) \\ &= \mathrm{TPR}\, p_U(y_1) + \mathrm{FPR}(1 - p_U(y_1)) \end{aligned}$$

- So we have:

$$p_U(y_1) = \frac{p_U^{CC}(y_1) - \mathrm{FPR}}{\mathrm{TPR} - \mathrm{FPR}} \tag{2}$$

$$p_U^{ACC}(y_1) = \frac{p_U^{CC}(y_1) - \hat{\mathrm{FPR}}_L}{\hat{\mathrm{TPR}}_L - \hat{\mathrm{FPR}}_L} \tag{3}$$

- However, the estimates could fall outside the range [0,1], in which case a clipping and rescaling is needed (something which is hardly justified from a theoretical point of view)
- Also, the hypothesis that $\hat{\mathrm{TPR}}_L$ is a good estimate of TPR, and $\hat{\mathrm{FPR}}_L$ is a good estimate of FPR, has been criticized.

Quantification methods: PACC

- Probabilistic Adjusted Classify and Count (PACC – Bella et al., 2010) stands to ACC like PCC stands to CC.
- Assuming y_1 is the positive class and y_2 the negative one, in the binary case this comes down to replacing:
  - $p_U^{CC}(y_i)$ with the expected counts $p_U^{PCC}(y_i)$
  - TPR with the expected counts $\frac{1}{|\{\mathbf{x}\in y_1\}|}\sum_{\mathbf{x}\in y_1}\Pr(y_1|\mathbf{x})$
  - FPR with the expected counts $\frac{1}{|\{\mathbf{x}\in y_2\}|}\sum_{\mathbf{x}\in y_2}\Pr(y_1|\mathbf{x})$
- Using a soft classifier s for estimating the posterior probabilities.
- TPR and FPR are estimated in L via cross-validation as before
- In the multiclass case, there is a system of $|\mathcal{Y}|$ linear equations with $|\mathcal{Y}|$ unknowns

A. Bella, C. Ferri, J. Hernańdez-Orallo, M.J. Ramírez-Quintana. Quantification via probability estimators. ICDM 2010.

Quantification methods: EMQ (cont'd)

- EMQ (Saerens et al., 2002): an EM-based class prevalence estimation method for improving classification accuracy
- EMQ consists of an iterative, mutually recursive re-computation of the posteriors $p(y|\mathbf{x})$ and of the priors $p_U(y)$, until convergence
- Method originally devised for improving the posteriors $p(y|\mathbf{x})$. But if quantification is our goal we can use its "byproducts", i.e., the improved estimates of the priors $p_U(y)$.

M. Saerens, P. Latinne, and C. Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation* 14(1):21–41, 2002.

Quantification methods: EMQ (cont'd)

- We apply EM in the following way until convergence of the $\hat{p}^{(s)}(y)$:
  - **Step 0**: For each $y \in \mathcal{Y}$ initialize $\hat{p}_U^{(0)}(y) \leftarrow p_L(y)$
    For each $\mathbf{x} \in U$ initialize $p^{(0)}(y|\mathbf{x}) \leftarrow p(y|\mathbf{x})$
  - **Step s**: Iterate $s = 1, 2, \ldots$ until convergence:
    - **Step s(E)**: For each y compute:

$$\hat{p}_U^{(s)}(y) = \frac{1}{|U|} \sum_{\mathbf{x} \in U} p^{(s-1)}(y|\mathbf{x}) \tag{4}$$

    - **Step s(M)**: For each unlabelled item $\mathbf{x}$ and each y compute:

$$p^{(s)}(y|\mathbf{x}) = \frac{\frac{\hat{p}_U^{(s)}(y)}{\hat{p}_U^{(0)}(y)} \cdot p^{(0)}(y|\mathbf{x})}{\sum_{y \in \mathcal{Y}} \frac{\hat{p}_U^{(s)}(y)}{\hat{p}_U^{(0)}(y)} \cdot p^{(0)}(y|\mathbf{x})} \tag{5}$$

- Step s(E) re-estimates the priors in terms of the new posterior probabilities
- Step s(M) re-calibrates the posterior probabilities by using the new priors

Quantification methods: EMQ (cont'd)

- We apply EM in the following way until convergence of the $\hat{p}^{(s)}(y)$:
  - **Step 0**: For each $y \in \mathcal{Y}$ initialize $\hat{p}_U^{(0)}(y) \leftarrow p_L(y)$
    For each $\mathbf{x} \in U$ initialize $p^{(0)}(y|\mathbf{x}) \leftarrow p(y|\mathbf{x})$
  - **Step s**: Iterate $s = 1, 2, \ldots$ until convergence:
    - **Step s(E)**: For each y compute:

$$\hat{p}_U^{(s)}(y) = \frac{1}{|U|} \sum_{\mathbf{x} \in U} p^{(s-1)}(y|\mathbf{x}) \tag{4}$$

    - **Step s(M)**: For each unlabelled item $\mathbf{x}$ and each y compute:

$$p^{(s)}(y|\mathbf{x}) = \frac{\frac{\hat{p}_U^{(s)}(y)}{\hat{p}_U^{(0)}(y)} \cdot p^{(0)}(y|\mathbf{x})}{\sum_{y \in \mathcal{Y}} \frac{\hat{p}_U^{(s)}(y)}{\hat{p}_U^{(0)}(y)} \cdot p^{(0)}(y|\mathbf{x})} \tag{5}$$

- Step s(E) re-estimates the priors in terms of the new posterior probabilities
- Step s(M) re-calibrates the posterior probabilities by using the new priors

Quantification methods: EMQ (cont'd)

- We apply EM in the following way until convergence of the $\hat{p}^{(s)}(y)$:
  - **Step 0**: For each $y \in \mathcal{Y}$ initialize $\hat{p}_U^{(0)}(y) \leftarrow p_L(y)$
    For each $\mathbf{x} \in U$ initialize $p^{(0)}(y|\mathbf{x}) \leftarrow p(y|\mathbf{x})$
  - **Step s**: Iterate $s = 1, 2, \ldots$ until convergence:
    - **Step s(E)**: For each y compute:

$$\hat{p}_U^{(s)}(y) = \frac{1}{|U|} \sum_{\mathbf{x} \in U} p^{(s-1)}(y|\mathbf{x}) \tag{4}$$

    - **Step s(M)**: For each unlabelled item $\mathbf{x}$ and each y compute:

$$p^{(s)}(y|\mathbf{x}) = \frac{\frac{\hat{p}_U^{(s)}(y)}{\hat{p}_U^{(0)}(y)} \cdot p^{(0)}(y|\mathbf{x})}{\sum_{y \in \mathcal{Y}} \frac{\hat{p}_U^{(s)}(y)}{\hat{p}_U^{(0)}(y)} \cdot p^{(0)}(y|\mathbf{x})} \tag{5}$$

- Step s(E) re-estimates the priors in terms of the new posterior probabilities
- Step s(M) re-calibrates the posterior probabilities by using the new priors

Quantification methods: EMQ (cont'd)

- We apply EM in the following way until convergence of the $\hat{p}^{(s)}(y)$:
  - **Step 0**: For each $y \in \mathcal{Y}$ initialize $\hat{p}_U^{(0)}(y) \leftarrow p_L(y)$
    For each $\mathbf{x} \in U$ initialize $p^{(0)}(y|\mathbf{x}) \leftarrow p(y|\mathbf{x})$
  - **Step s**: Iterate $s = 1, 2, \ldots$ until convergence:
    - **Step s(E)**: For each y compute:

$$\hat{p}_U^{(s)}(y) = \frac{1}{|U|} \sum_{\mathbf{x} \in U} p^{(s-1)}(y|\mathbf{x}) \tag{4}$$

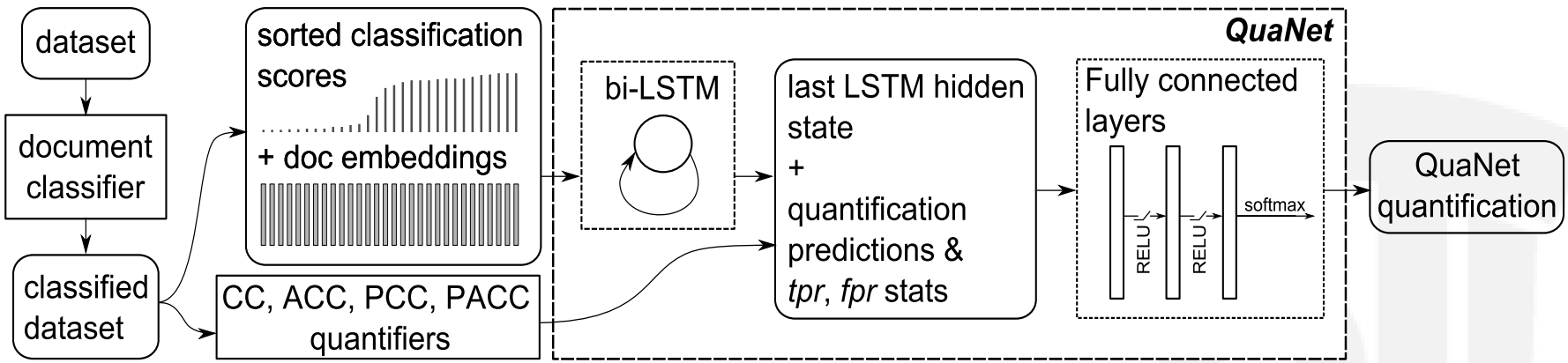
    - **Step s(M)**: For each unlabelled item $\mathbf{x}$ and each y compute:

$$p^{(s)}(y|\mathbf{x}) = \frac{\frac{\hat{p}_U^{(s)}(y)}{\hat{p}_U^{(0)}(y)} \cdot p^{(0)}(y|\mathbf{x})}{\sum_{y \in \mathcal{Y}} \frac{\hat{p}_U^{(s)}(y)}{\hat{p}_U^{(0)}(y)} \cdot p^{(0)}(y|\mathbf{x})} \tag{5}$$

- Step s(E) re-estimates the priors in terms of the new posterior probabilities
- Step s(M) re-calibrates the posterior probabilities by using the new priors

Quantification methods: QuaNet

- QuaNet (Esuli et al., 2018) is a deep learning -based method for quantification
- The idea is to learn to produce higher-order quantification embeddings, i.e., an embedded representation of U from the:
  - observed posterior probabilites generated by a classifier
  - document embeddings
  - quantification predictions of simple aggregative methods
- QuaNet is trained across the full prevalence spectrum in order to learn how to adjust the counts it receives



A. Esuli, A. Moreo, and F. Sebastiani. A recurrent neural network for sentiment quantification. CIKM 2018.

Other quantification methods

- The Mixture Model re-parameterizes a mixture model of class-conditional posterior probabilities from L so that the target distribution fits best the validation distribution: MM (Forman 2005) HDy, DyS, SSM
- T50, Method X, Method Max, Median Sweep: different heuristics to choose the classification threshold to improve the stability of the denominator of ACC (Forman 2006)
- ReadMe is an example of non-aggregative quantification, in which the system of equations is framed as $\Pr(X) = \Pr(X|Y)\Pr(Y)$.
- Explicit loss minimization methods use a learner which directly optimizes the evaluation measure ("loss") used for quantification (Esuli and Sebastiani, 2010)

Forman, G., Counting Positives Accurately Despite Inaccurate Classification. ECML 2005.

Forman, G., Quantifying trends accurately despite classifier error and class imbalance. KDD 2006.

D. Hopkins and G. King (2010). A method of automated nonparametric content analysis for social science. *American Journal of Political Science*: 54(1), 229–247.

A. Esuli and F. Sebastiani. Sentiment quantification. *IEEE Intelligent Systems* 25(4):72–75, 2010.

Structure of this lecture

1. Introduction
2. Applications of quantification in IR, ML, DM, NLP
3. Evaluation measures for quantification
4. Supervised learning methods for quantification
5. Advanced topics
6. Resources and shared tasks
7. Conclusions

Advanced topics (hints)

- Ordinal quantification (Bunse et al., 2022)
- Estimating Fairness (Fabris et al., 2022)
- Regression quantification (Bella et al., 2014)
- Cross-lingual text quantification (Moreo et al., 2019)
- Quantification for data streams (Maletzke et al., 2018)
- Quantification for networked data (Tang et al., 2010)
- ...

Structure of this lecture

1. Introduction
2. Applications of quantification in IR, ML, DM, NLP
3. Evaluation measures for quantification
4. Supervised learning methods for quantification
5. Advanced topics
6. Resources and shared tasks
7. Conclusions

Software resources for quantification

QuaPy is an open source framework for quantification written in Python. Available in GitHub (https://github.com/HLT-ISTI/QuaPy).

```python
import quapy as qp
from quapy.method.aggregative import PACC
from sklearn.linear_model import LogisticRegression
import numpy as np
import pandas as pd

# setting this environment variable allows some
# error metrics (e.g., mrae) to be smoothed
qp.environ["SAMPLE_SIZE"] = 500

dataset = qp.datasets.fetch_reviews('imdb', tfidf=True, min_df=5)

# model selection with the APP
model = qp.model_selection.GridSearchQ(
 model=PACC(LogisticRegression()),
 param_grid={
 'C': np.logspace(-4, 5, 10),
 'class_weight': ['balanced', None]
 },
 protocol='app',
 eval_budget=100,
 error='mae',
 refit=True, # retrain on the whole labelled set once done
 val_split=0.25,
).fit(dataset.training)

df = qp.evaluation.artificial_prevalence_report(
 model, # the quantification method
 dataset.test, # the set on which the method will be evaluated
 n_prevpoints=101, # i.e., using the grid [0.,.01,.02,...,.99,1.]
 n_jobs=-1, # the number of parallel workers (-1 for all CPUs)
 random_seed=42, # allows replicating test samples across runs
 error_metrics=['ae', 'rae', 'kld']) # evaluation metrics

print(f'best hyper-params={model.best_params_}')

pd.set_option('display.max_columns', None)
pd.set_option('display.width', 100)
print(df)
```

Moreo, A., Esuli, A., and Sebastiani, F. QuaPy: A Python-based frame work for quantification. In Proceedings of the 30th ACM International Conference on Knowledge Management (CIKM 2021), pages 4534–4543, Gold Coast, AU.

Structure of this lecture

1. Introduction
2. Applications of quantification in IR, ML, DM, NLP
3. Evaluation measures for quantification
4. Supervised learning methods for quantification
5. Advanced topics
6. Resources and shared tasks
7. Conclusions

Conclusion

- Quantification: a relatively (yet) unexplored task, with many research problems still open
- Growing awareness that quantification is going to be more and more important; given the advent of big data, more and more application contexts will spring up in which we will simply be happy with analysing data at the aggregate (rather than at the individual) level.
- If you are interested in quantification:
  - LQ2: workshop at ECML2022, Grenoble (Fr), 23rd Sep.
  - LeQua: shared task at CLEF2022, Bologna (It), 5-8 Sep.
  - Learning to Quantify: book at Springer (forthcoming end of 2022).

Thank you!

Questions?

For any question:

alejandro.moreo@isti.cnr.it

and

fabrizio.sebastiani@isti.cnr.it