

# Machine Learning for Automatic Text Analysis: Tasks, Methods, and Tools

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

Summer School “Digital Tools for Humanists”  
Pisa, IT – June 7-16, 2022

Version 1.0

Download most recent version of these slides at  
<https://bit.ly/3zq6QXX>

# Introduction

- Many text analysis tasks are either tedious / expensive / time-consuming / difficult to carry out; e.g.,
  - Coding textual answers returned to open questions in questionnaires (e.g., in opinion research, market research, customer relationship management)
  - Marking a textual comment (on a product, on a political candidate, etc.) as conveying a positive or a negative opinion about its subject
  - Detecting whether a suspect (e.g., John) is the author of an anonymous text
  - Checking scientific papers for inclusion in a “systematic review”
  - Choosing the most competent examiner for a given patent application
  - Assigning subject codes (from a predefined taxonomy) to scientific papers
- Can these tasks be automated to some degree?
- Can we build tools that support the work of humans who carry out these tasks?

# Part I :

# Text Classification



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



# Introduction

- Coding textual answers returned to open questions in questionnaires (e.g., in opinion research, market research, customer relationship management)  
→ 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  
→ codeframe is {Positive,Negative,Neutral} (**Sentiment Classification**)
- Detecting whether a suspect (e.g., John) is the author of an anonymous text  
→ codeframe is {John,NotJohn} (**Authorship Verification**)
- Checking scientific papers for inclusion in a “systematic review”  
→ codeframe is {Include,DontInclude}
- Choosing the most competent examiner for a given patent application  
→ codeframe is the set of available examiners
- Assigning subject codes (from a predefined taxonomy) to scientific papers  
→ codeframe is the taxonomy of subject codes

# Introduction

- Can classification be automated?
- Can we build tools that support the work of humans who need to classify text?
- Problems:
  - unlike other types of data (e.g., factual data, numerical measurements, etc.), text requires (subjective) interpretation
    - 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

- 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 short 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

- 1 Text Classification
- 2 Applications of Text Classification
- 3 Supervised Learning and Text Classification
  - 1 Representing Text for Classification Purposes
  - 2 Training a Classifier
- 4 Evaluating a Classifier
- 5 Advanced Topics (Hints)





# 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

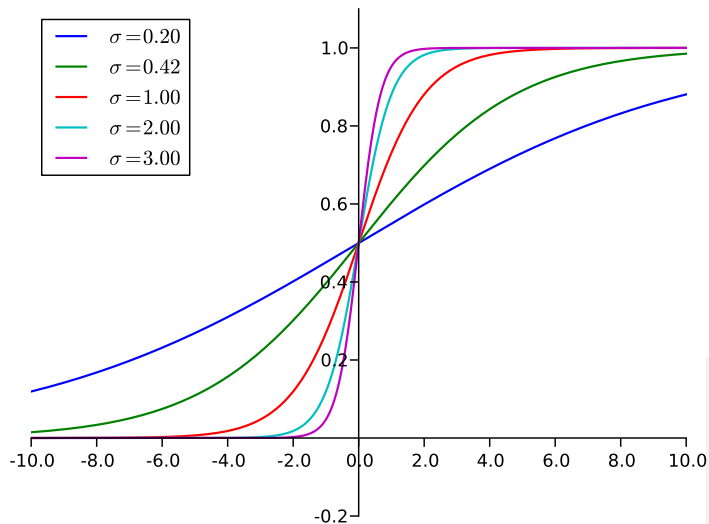
- **Binary** classification:  $h : \mathcal{D} \rightarrow \mathcal{C}$  (each item belongs to exactly one class) and  $\mathcal{C} = \{c_1, c_2\}$ 
  - E.g., assigning emails to one of {Spam, Legitimate}
- **Single-Label Multi-Class** (SLMC) classification:  $h : \mathcal{D} \rightarrow \mathcal{C}$  (each item belongs to exactly one class) and  $\mathcal{C} = \{c_1, \dots, c_n\}$ , 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  $\mathcal{C} = \{c_1, \dots, c_n\}$ , 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  $c_1 \preceq \dots \preceq c_n$  on  $\mathcal{C} = \{c_1, \dots, c_n\}$ 
  - 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  $(d, c)$ , where the score denotes the { probability / strength of evidence / confidence } that  $d$  belongs to  $c$ 
  - E.g., a probabilistic classifier outputs “posterior probabilities”  $\Pr(c|d) \in [0, 1]$
  - E.g., the AdaBoost classifier outputs scores  $s(d, c) \in (-\infty, +\infty)$  that represent its confidence that  $d$  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**:

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

# 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  $s(d, c)$
  - ② Picking a **threshold**  $t$ , such that
    - $s(d, c) \geq t$  is interpreted as predicting  $c_1$
    - $s(d, c) < t$  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 language** (a.k.a. “language identification”); useful, e.g., in query processing within search engines
- **by genre** ; e.g., AutomotiveNews vs. AutomotiveBlogs, useful in website classification and others;
- **by author** (a.k.a. “authorship attribution”), or **by native language** (“native language identification”); useful in philology, forensics, and cybersecurity
- ...

# Text Classification

- 1 Text Classification
- 2 Applications of Text Classification
- 3 Supervised Learning and Text Classification
  - 1 Representing Text for Classification Purposes
  - 2 Training a Classifier
- 4 Evaluating a Classifier
- 5 Advanced Topics (Hints)



## Example 1: Knowledge Organization

- Long tradition in both science and the humanities ; goal was organizing knowledge, i.e., conferring structure to an otherwise unstructured body of knowledge
- The rationale is that using a structured body of knowledge is easier / more effective than if this knowledge is unstructured
- **Automated** classification tries to automate the tedious task of assigning data items based on their content, a task otherwise performed by human annotators (a.k.a. “assessors”, or “coders”)



## Example 1: Knowledge Organization (cont'd)

- Scores of applications; e.g.,
  - Classifying news articles for selective dissemination
  - Classifying scientific papers into specialized taxonomies
  - Classifying patents
  - Classifying “classified” ads
  - Classifying answers to open-ended questions
  - Classifying topic-related tweets by sentiment
  - ...
- Retrieval (as in search engines) could also be viewed as (binary + soft) classification into Relevant vs. NonRelevant

## Example 2: Filtering

- **Filtering** (a.k.a. “routing”) using refers to the activity of blocking a set of NonRelevant items from a dynamic stream, thereby leaving only the Relevant ones
  - E.g., **spam filtering** is an important example, attempting to tell Legitimate messages from Spam messages<sup>1</sup>
  - **Detecting unsuitable content** (e.g., porn, violent content, racist content, cyberbullying, fake news) is also an important application, e.g., in PG filters or on interfaces to social media
- Filtering is thus an instance of binary (usually: hard) classification, and its applications are ubiquitous

---

<sup>1</sup>Gordon V. Cormack: Email Spam Filtering: A Systematic Review. *Foundations and Trends in Information Retrieval* 1(4):335–455 (2006)

## Example 3: Empowering other IR Tasks

- Functional to improving the effectiveness of other tasks in IR or NLP; e.g.,
  - Classifying queries by intent within search engines
  - Classifying questions by type in question answering systems
  - Classifying named entities
  - Word sense disambiguation in NLP systems
  - ...
- Many of these tasks involve classifying very small texts (e.g., queries, questions, sentences), and stretch the notion of “text” classification quite a bit ...

# Text Classification

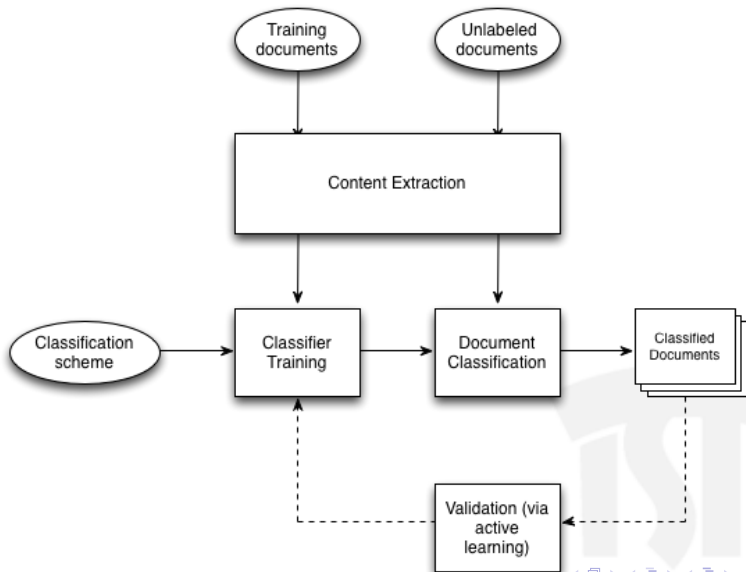
- 1 Text Classification
- 2 Applications of Text Classification
- 3 **Supervised Learning and Text Classification**
  - 1 Representing Text for Classification Purposes
  - 2 Training a Classifier
- 4 Evaluating a Classifier
- 5 Advanced Topics (Hints)



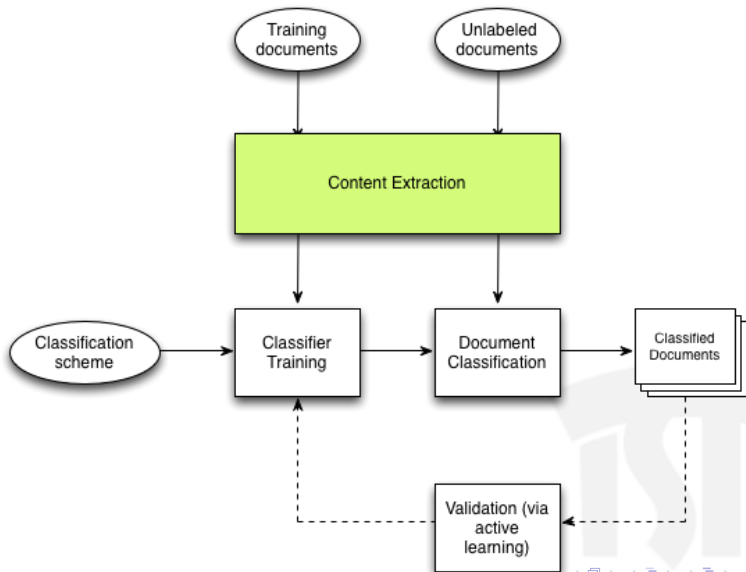
# The Supervised Learning Approach to Classification

- An old-fashioned way to build text classifiers was via **knowledge engineering**, i.e., manually building classification rules
  - E.g., (Viagra **or** Sildenafil **or** Cialis) → Spam
- Disadvantages: expensive to setup and to maintain
- Superseded by the **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



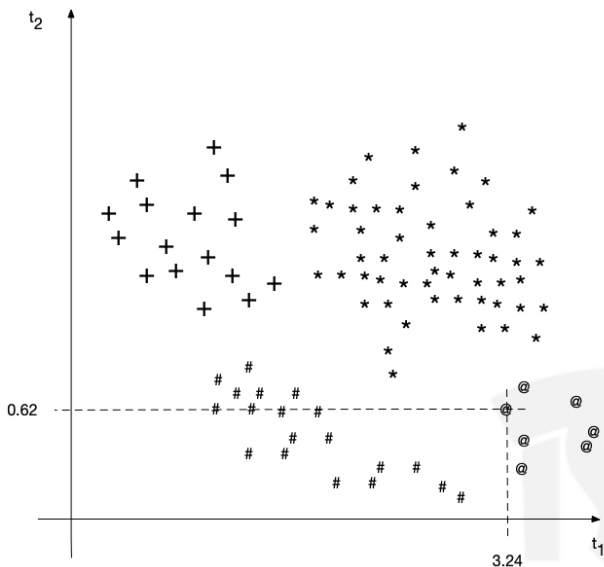
# Representing Text for Classification Purposes

- 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

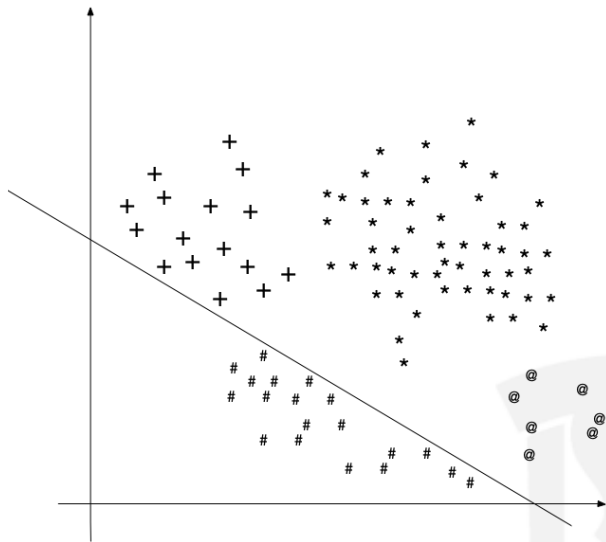




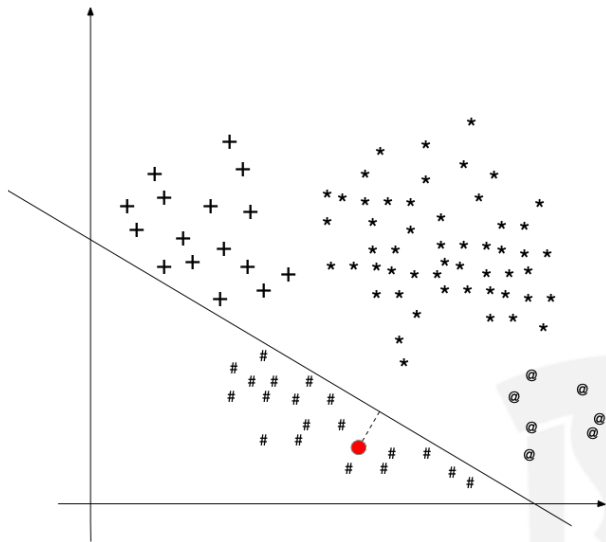
# Representing Text for Classification Purposes



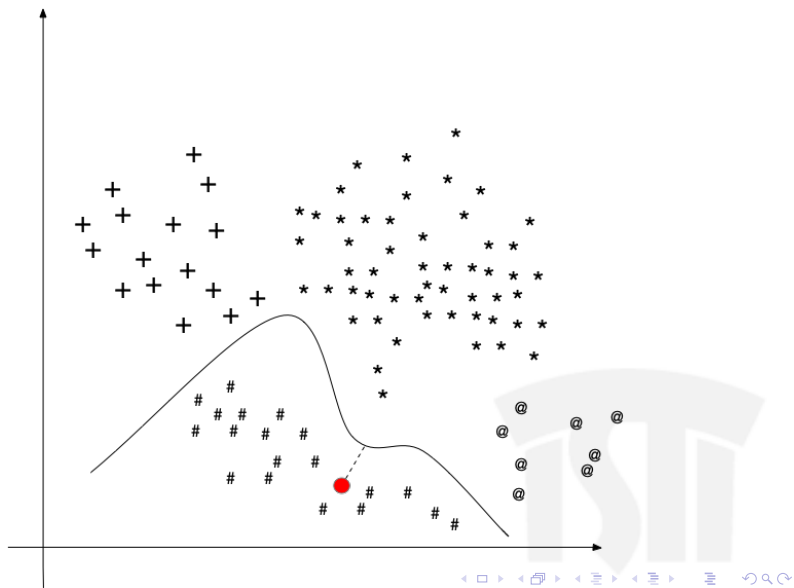
# Representing Text for Classification Purposes



# Representing Text for Classification Purposes



# Representing Text for Classification Purposes



# Representing Text 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: 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

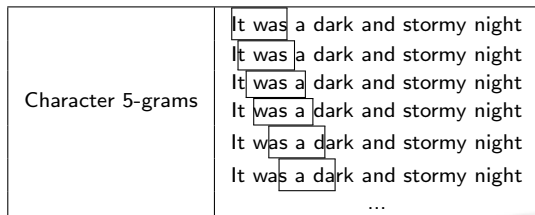
- Word  $n$ -grams (i.e., sequences of  $n$  words that frequently occur in  $L$ ) may be optionally added; this is usually limited to  $n = 2$  (**unigram+bigram model**)

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)<sup>2</sup>



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

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



# Representing Text: 1. Feature Extraction

- The above is OK for classification by topic, but not necessarily when classifying by other dimensions!
- E.g.
  - in classification by author, features such average word length, average sentence length, punctuation frequency, frequency of subjunctive clauses, etc., are used<sup>3</sup>

---

<sup>3</sup>Patrick Juola: Authorship Attribution. *Foundations and Trends in Information Retrieval* 1(3): 233-334 (2006)

# Representing Text: 1. Feature Extraction

- The above is OK for classification by topic, but not necessarily when classifying by other dimensions!
- E.g.
  - in classification by author, features such average word length, average sentence length, punctuation frequency, frequency of subjunctive clauses, etc., are used<sup>3</sup>

---

<sup>3</sup>Patrick Juola: Authorship Attribution. *Foundations and Trends in Information Retrieval* 1(3): 233-334 (2006)

# Representing Text: 1. Feature Extraction

- The above is OK for classification by topic, but not necessarily when classifying by other dimensions!
- E.g.
  - in classification by author, features such average word length, average sentence length, punctuation frequency, frequency of subjunctive clauses, etc., are used<sup>3</sup>
  - In classification by sentiment, bag-of-words is not enough, and deeper linguistic processing is necessary
- The choice of features for a classification task (**feature design**) is dictated by the distinctions we want to capture, and is left to the designer.

---

<sup>3</sup>Patrick Juola: Authorship Attribution. *Foundations and Trends in Information Retrieval* 1(3): 233-334 (2006)

## 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  $\xi$ ) the discriminative power  $\xi(t_k)$  of each feature  $t_k$  and retaining only the top-scoring features<sup>4</sup>
- For binary classification, a typical choice for  $\xi$  is **mutual information**, i.e.,

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

Alternative choices are chi-square and log-odds.

---

<sup>4</sup>Y. Yang, J. Pedersen: A Comparative Study on Feature Selection in Text Categorization. Proceedings of ICML 1997.

## Representing Text: 2b. Feature Synthesis

- **Matrix decomposition techniques** (e.g., PCA, SVD, LSA) can be used to synthesize new features that replace the features discussed above with ones not suffering from **ambiguity** and **polisemy**
- These techniques are based on the principles of **distributional semantics**, which states that the semantics of a word “is” the words it co-occurs with in corpora of language use

*You shall know a word by the company it keeps*

(John R. Firth, 1957)

- Pros: the synthetic features in the new vectorial representation do not suffer from polisemy or synonymy
- Cons: computationally expensive, sometimes prohibitively so
- **Word embeddings**: the “new wave of distributional semantics”, as from “deep learning”<sup>5</sup>

---

<sup>5</sup>Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; and Dean, J. Distributed representations of words and phrases and their compositionality. NIPS, 2013.

## Representing Text: 3. Feature Weighting

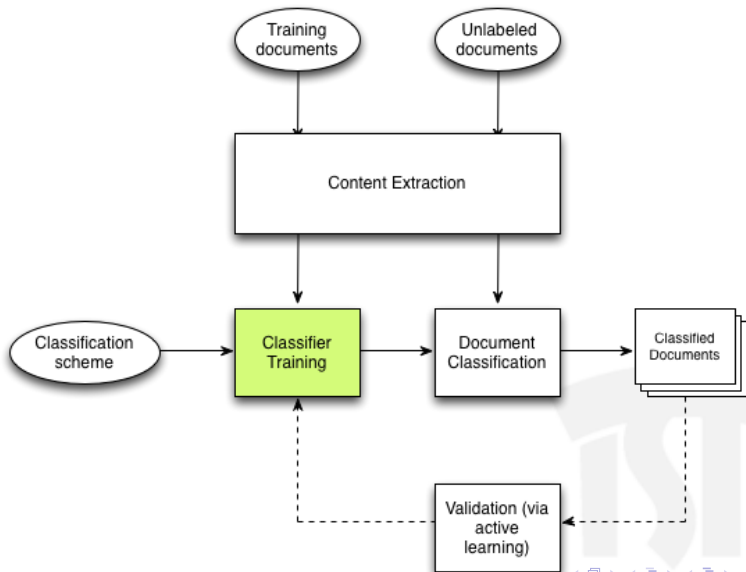
- **Feature weighting** means attributing a value  $x_{ik}$  to feature  $t_k$  in the vector  $\mathbf{x}_i$  that represents document  $d_i$ ; this value may be
  - **binary** (representing presence/absence of  $t_k$  in  $d_i$ ); or
  - **numeric** (representing the importance of  $t_k$  for  $d_i$ ); 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 similarity between two vectors may be computed via **cosine similarity**

$$sim(\mathbf{x}_1, \mathbf{x}_2) = \frac{\sum_{i=1}^K x_{i1} \cdot x_{i2}}{(\sum_{i=1}^K x_{i1}^2)^{\frac{1}{2}} (\sum_{i=1}^K x_{i2}^2)^{\frac{1}{2}}}$$

If these vectors are pre-normalized, this is equivalent to computing their **dot product**

$$sim(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=1}^K x_{i1} \cdot x_{i2}$$

# The Supervised Learning Approach to Classification



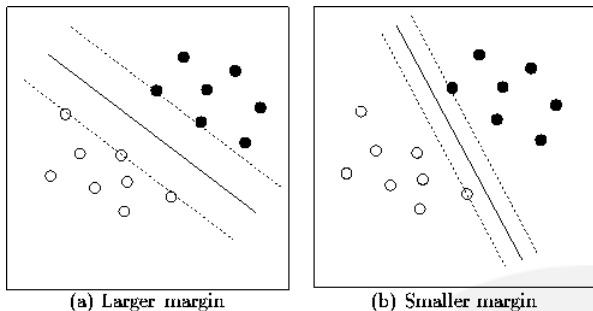
# Supervised Learning for Binary Classification

- 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



# An Example Supervised Learning Method: SVMs

- A **constrained optimization problem**: find the separating surface (e.g., hyperplane) that maximizes the **margin** (i.e., the minimum distance between the hyperplane and the training examples)



- Margin maximization conducive to good generalization accuracy on unseen data
- Theoretically well-founded + good empirical performance on a variety of tasks
- Publicly available implementations optimized for high-dimensional, sparse feature spaces: e.g., SVM-Light, LibSVM, LibLinear,

# An Example Supervised Learning Method: SVMs (cont'd)

- We consider linear separators (i.e., hyperplanes) and classifiers of type

$$h(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

- **Hard-margin SVMs** look for

$$\begin{array}{ll} \arg \min_{\mathbf{w} \geq 0} & \frac{1}{2} \mathbf{w} \cdot \mathbf{w} \\ \text{such that} & y_i [\mathbf{w} \cdot \mathbf{x}_i + b] \geq 0 \\ & \text{for all } i \in \{1, \dots, |L|\} \end{array}$$

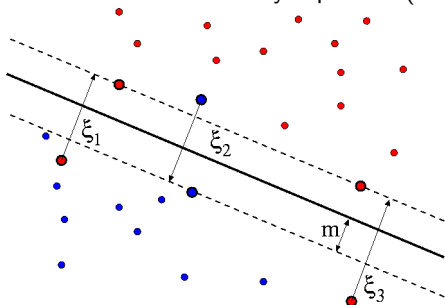
- There are now fast algorithms for this<sup>6</sup>

---

<sup>6</sup>T. Joachims, C.-N. Yu: Sparse kernel SVMs via cutting-plane training. *Machine Learning*, 2009.

# An Example Supervised Learning Method: SVMs (cont'd)

- Classification problems are often not linearly separable (LS)

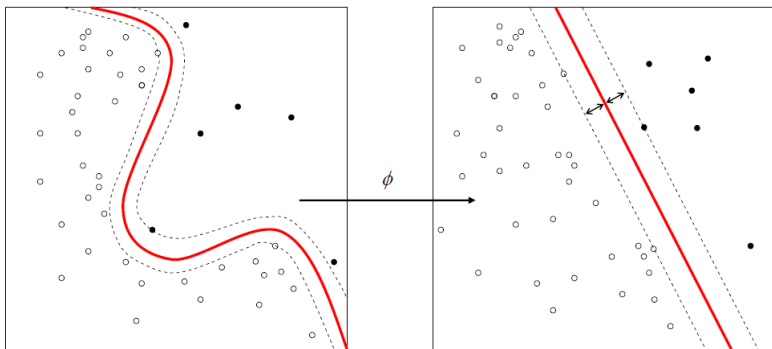


- Soft-margin SVMs** introduce penalties for misclassified training examples; they look for

$$\begin{aligned} \arg \min_{\mathbf{w}, \xi_i \geq 0} \quad & \frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_{i=1}^{|\mathcal{L}|} \xi_i \\ \text{such that} \quad & y_i' [\mathbf{w} \cdot \mathbf{x}_i' + b] \geq (1 - \xi_i) \\ & \text{for all } i \in \{1, \dots, |\mathcal{L}|\} \end{aligned}$$

# An Example Supervised Learning Method: SVMs (cont'd)

- Non-LS problems can become LS once mapped to a high-dimensional space



## An Example Supervised Learning Method: SVMs (cont'd)

- **Kernels** are similarity functions  $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$ , where  $\phi(\cdot)$  is a mapping into a higher-dimensional space
- SVMs can indeed use kernels instead of the standard dot product; popular kernels are
  - $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$  (the linear kernel)
  - $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i \cdot \mathbf{x}_j + r)^d, \gamma > 0$  (the polynomial kernel)
  - $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \gamma > 0$  (the RBF kernel)
  - $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i \cdot \mathbf{x}_j + r)$  (the sigmoid kernel)
- However, the linear kernel is usually employed in text classification applications; there are theoretical arguments supporting this<sup>7</sup>.

---


<sup>7</sup>T. Joachims: A Statistical Learning Model of Text Classification for Support Vector Machines. Proceedings of SIGIR 2001.

# 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<sup>8</sup>
- For **ordinal** classification, algorithms customised to OC need to be used (e.g., SVORIM, SVOREX)<sup>9</sup>

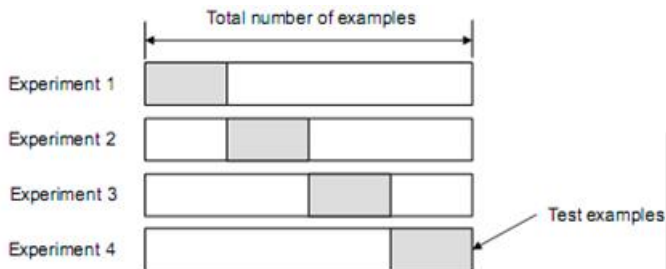
---

<sup>8</sup>K. Crammer and Y. Singer. On the Algorithmic Implementation of Multi-class SVMs, *Journal of Machine Learning Research*, 2001.

<sup>9</sup>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  $\gamma$ ,  $r$ ,  $d$  parameters of non-linear kernels
  - ...
- These parameters need to be optimized, e.g., via  **$k$ -fold cross-validation** on the training set



# Text Classification

- 1 Text Classification
- 2 Applications of Text Classification
- 3 Supervised Learning and Text Classification
  - 1 Representing Text for Classification Purposes
  - 2 Training a Classifier
- 4 Evaluating a Classifier
- 5 Advanced Topics (Hints)





# 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  $\Omega$

		true	
		YES	NO
pred	YES	TP	FP
	NO	FN	TN

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

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

- $F_1$  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
  - micro-averaging**: compute  $F_1$  from the “collective” contingency table obtained by summing cells

		true	
		YES	NO
predicted	YES	$\sum_{c_j \in \mathcal{C}} TP_i$	$\sum_{c_j \in \mathcal{C}} FP_i$
	NO	$\sum_{c_j \in \mathcal{C}} FN_i$	$\sum_{c_j \in \mathcal{C}} TN_i$

- macro-averaging**: compute  $F_1(c_j)$  for all  $c_j \in \mathcal{C}$  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  $\Omega_{ij}$  is the number of documents in  $c_i$  which are predicted to be in  $c_j$

		true		
		$c_1$	...	$c_{ C }$
pred	$c_1$	$\Omega_{11}$	...	$\Omega_{1 C }$
	...	...	...	...
	$c_{ C }$	$\Omega_{ C 1}$	...	$\Omega_{ C  C }$

- 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 one is **macroaveraged mean absolute error**, i.e.,

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

- 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.,

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

## Some Datasets for Evaluating Text Classification

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

- 1 Text Classification
- 2 Applications of Text Classification
- 3 Supervised Learning and Text Classification
  - 1 Representing Text for Classification Purposes
  - 2 Training a Classifier
- 4 Evaluating a Classifier
- 5 **Advanced Topics (Hints)**



## Advanced Topics (hints)

- Hierarchical classification
  - Classification when the classification scheme has a hierarchical nature
- Hypertext classification (an application of “Relational Learning”)
  - Classification when the items are hypertextual (e.g., Web pages)
- Cost-sensitive classification
  - 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

## Advanced Topics (hint)

- Cross-lingual text classification
  - Learning to classify documents in a language  $L_t$  from training data expressed in a language  $L_s$
- Semi-automated text classification
  - Optimizing the work of human assessors that need to review the results of automated classification
- Text quantification
  - Learning to estimate the distribution of the classes within the unlabelled data
- 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.
  - T. Joachims: *Learning to Classify Text using Support Vector Machines*. Kluwer, 2002.
- 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.

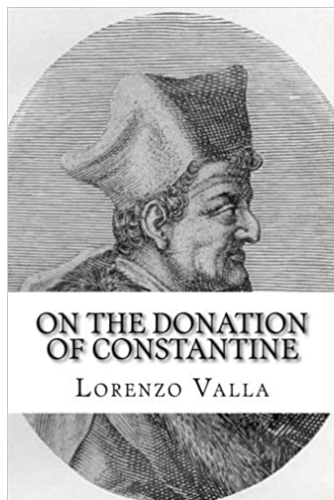
# Part II :

# 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 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**)
- Q: Can we automate NLI?
- A: **Yes.**

The screenshot shows a news article from 'la Repubblica' with the headline 'Bologna, la prof albanese corregge Salvini: "Quelle minacce non sono nostre, ecco perché"'. The article text reads: 'Il ministro dell'Interno aveva pubblicato sui social una lettera sgrammaticata i stranieri. Ma la docente, che insegna nel dipartimento di Lingue dell'Alma Mat facciamo quel tipo di errori grammaticali'. Below the article is a social media post from Matteo Salvini with a photo of a handwritten note that says 'PER ITALIA SAL'. The article is dated 05 dicembre 2018 and includes social media sharing icons for Facebook, Twitter, LinkedIn, and WhatsApp.

# (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 is 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 native language (L1) of the author of the text;
    - **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.

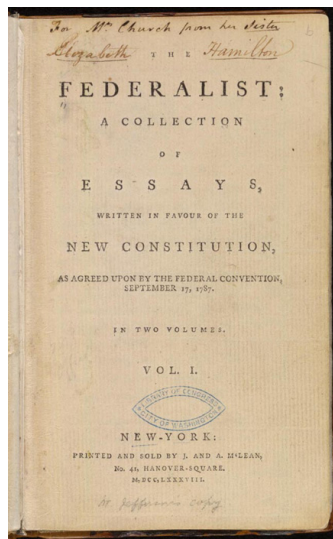
# (Computational) Authorship Analysis

- 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



# 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 its 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
  - function words
  - POS tags
  - token/word ratio
  - ...
- 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)



# 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  $\mathcal{A} = \{a_1, \dots, a_n\}$  (**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	<b>11</b>	<b>1</b>
FP	0	2
FN	1	1
TN	<b>282</b>	<b>26</b>
Total	294	30



# References

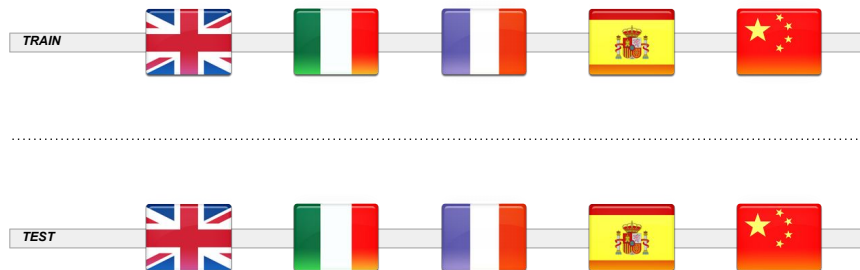
- Silvia Corbara, Alejandro Moreo, Fabrizio Sebastiani, and Mirko Tavoni. L'Epistola a Cangrande al vaglio della Computational Authorship Verification: Risultati preliminari (con una postilla sulla cosiddetta "XIV Epistola di Dante Alighieri"). In Alberto Casadei (ed.), *Atti del Seminario "Nuove Inchieste sull'Epistola a Cangrande"*, Pisa University Press, Pisa, IT, 153-192, 2020.  
<http://nmis.isti.cnr.it/sebastiani/Publications/Cangrande2020.pdf>
- 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/10.1561/15000000005>
- 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 III :

## Cross-Lingual Text Classification

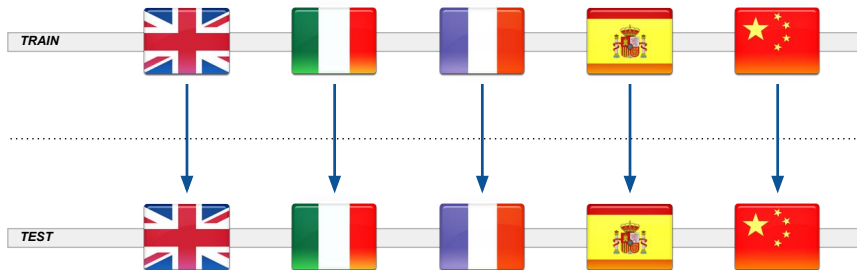


# Multilingual Text Classification



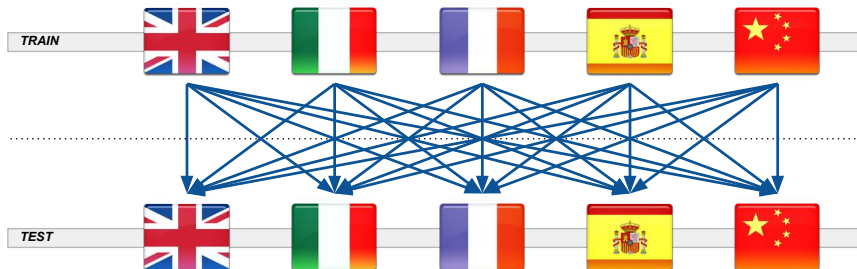
- Each document  $d$  written in one of a finite set  $\mathcal{L} = \{\lambda_1, \dots, \lambda_m\}$
- Codeframe  $\mathcal{C} = \{c_1, \dots, c_n\}$  is the same for all languages
- Scenario common in many multinational organizations / companies and in many multilingual countries
- Three “variants”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

# 1. (Multiple) Mono-lingual Text Classification



- MLC solved as  $m$  independent monolingual classification tasks

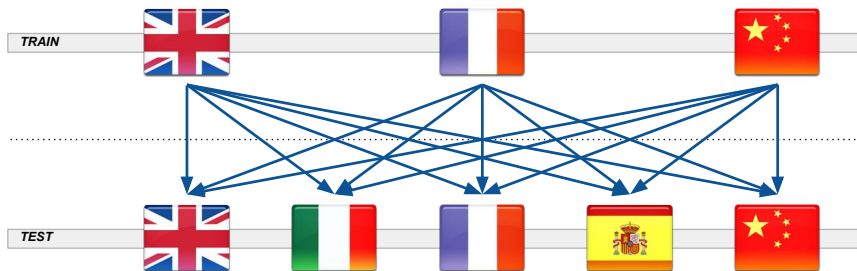
## 2. Poly-lingual Text Classification



- 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

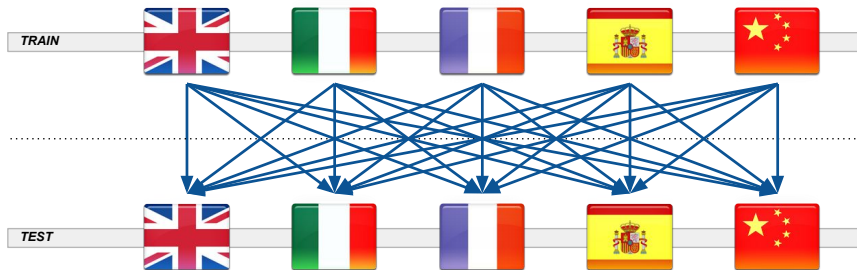


### 3. Cross-lingual Text Classification



- 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

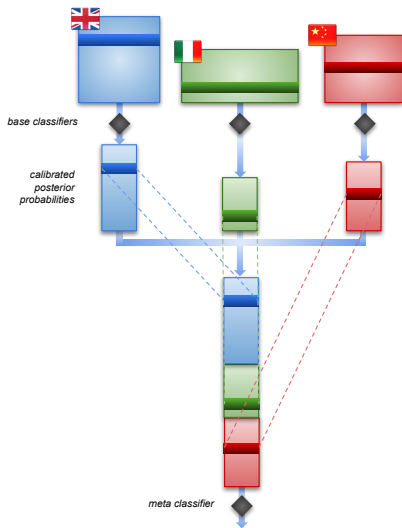
- PLC and CLC are instances of **(Heterogeneous) Transfer Learning (TL)**
- Basic idea of TL: reuse info about a problem in a **source** domain for solving the same problem in a different **target** domain
- CLC / PLC : problem = classification in  $\mathcal{C}$   
info = training examples  
domain = 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

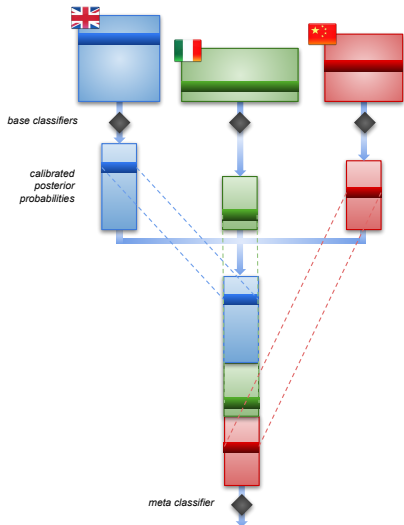


# Funnelling: PLC made easy



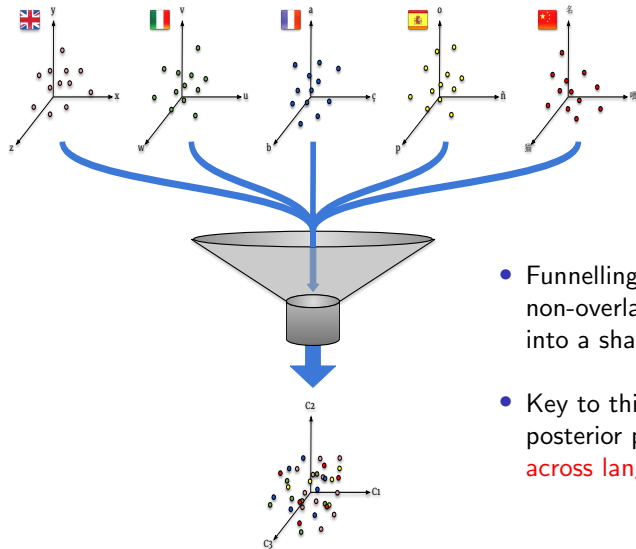
- Two-level classification architecture
  - 1 Set of language-dependent **base classifiers**
  - 2 Language-independent **metaclassifier**
- For the metaclassifier, document  $d$  represented as **vector of  $n$  classification scores**
- Metaclassifier outputs a vector of  $n$  classification scores

# 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

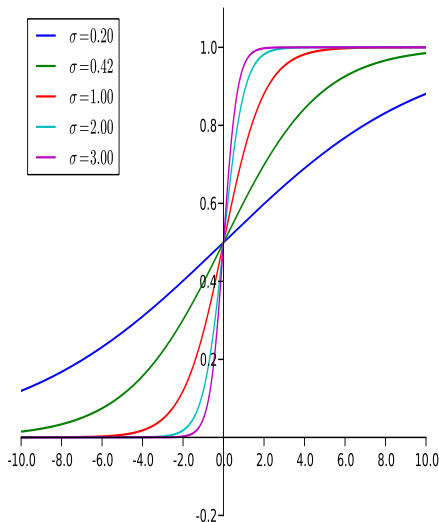
# Funnelling: PLC made easy



- 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

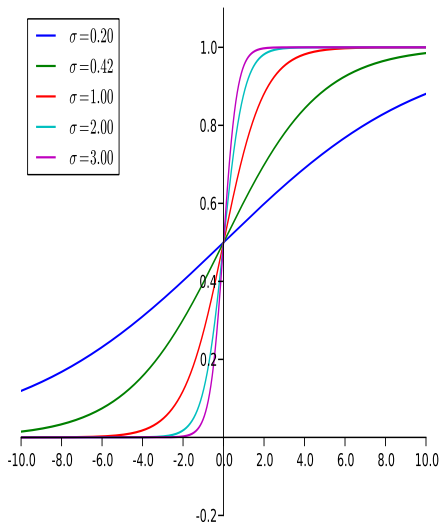




# Probability calibration

- Several calibration methods available off-the-shelf (e.g., “Platt calibration”)
- Needed for some learners and not for others; e.g.,

	Outputs Posterior Probs	Outputs WC Posterior Probs
SVMs	No	No
AdaBoost	No	No
Naive Bayes	Yes	No
Logistic Reg	Yes	Yes



# Training a funnelling system

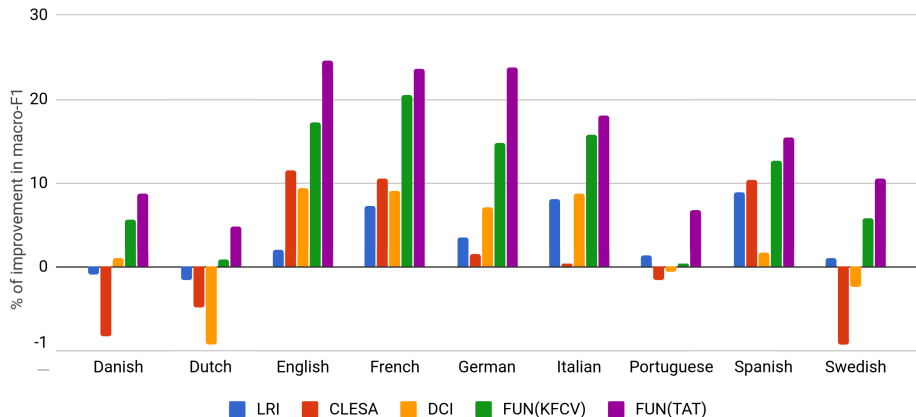
- 1 Train base classifiers using monolingual training sets
  - 2 Classify training examples
    - via trained classifiers (**Fun(TAT)**)
    - via  $k$ -fold cross-validation (**Fun(kFCV)**)
  - 3 Map classification scores into well-calibrated posterior probabilities
  - 4 Use posterior probabilities 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
  - → Choose via experimentation

# 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

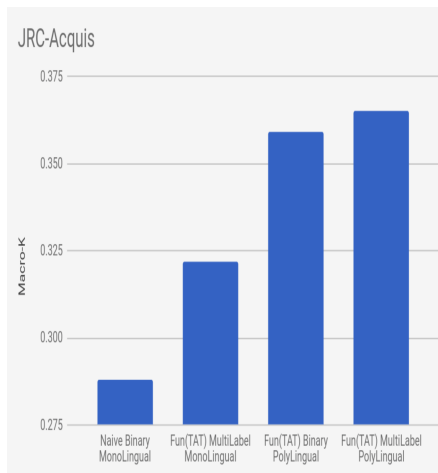
		NAÏVE	LRI	CLESA	DCI	FUN(KFCV)	FUN(TAT)
$F_1^\mu$	RCV1/RCV2	.776	.771	.714	.770	.801 <sup>†</sup>	<b>.802</b>
	JRC-Acquis	.559	<b>.594</b>	.557	.510	.581	.587
$F_1^M$	RCV1/RCV2	.467	.490	.471	.485	.512	<b>.534</b>
	JRC-Acquis	.340	<b>.411</b>	.379	.317	.356	.399
$K^\mu$	RCV1/RCV2	.690	.696	.659	.696	.731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

# Single-label PLC results

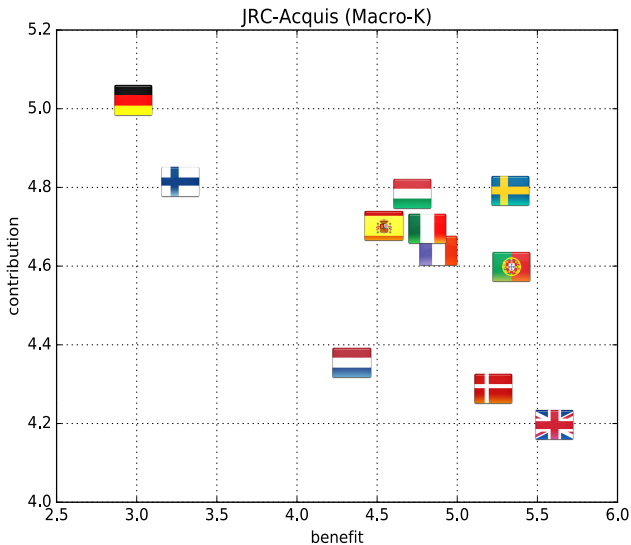
		NAÏVE	LRI	CLESA	DCI	FUN(KFCV)	FUN(TAT)
$F_1^\mu$	RCV1/RCV2	.759	.766	.706	.736	<b>.792</b>	.781
	JRC-Acquis	.202	<b>.353</b>	.331	.262	.318	.340 <sup>†</sup>
$F_1^M$	RCV1/RCV2	.538	.558	.543	.543	.584	<b>.596</b>
	JRC-Acquis	.362	<b>.407</b>	.400	.374	.382	.389
$K^\mu$	RCV1/RCV2	.649	.670	.636	.646	.715	<b>.757</b>
	JRC-Acquis	.115	.222	.215	.163	.205	<b>.253</b>
$K^M$	RCV1/RCV2	.503	.522	.521	.527	.559	<b>.594</b>
	JRC-Acquis	.358	.400	.396	.380	.389	<b>.407</b>

# 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?



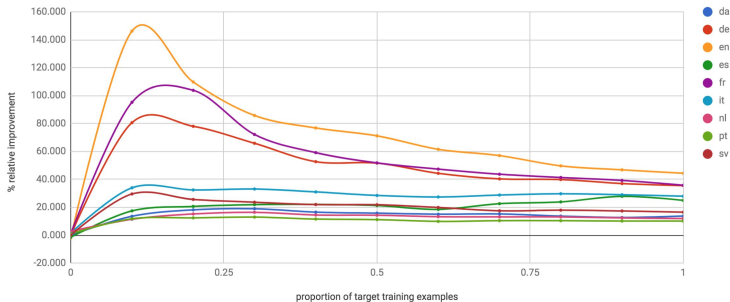
# Which languages benefit / contribute most?



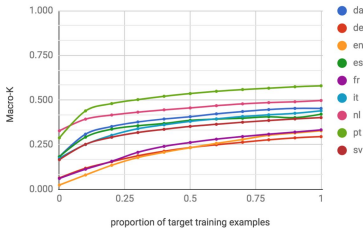


# How does this contribution evolve?

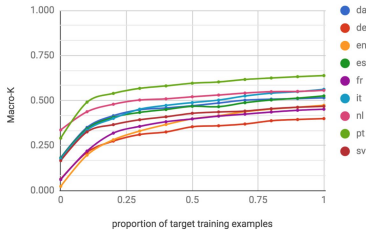
Cross-lingual relative improvement (Fun(TAT) vs. Naive) in RCV1/2



Performance of Naive in RCV1/2



Performance of Fun(TAT) in RCV1/2



# Part IV :

# Sentiment Analysis



# Sentiment Analysis and Opinion Mining

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

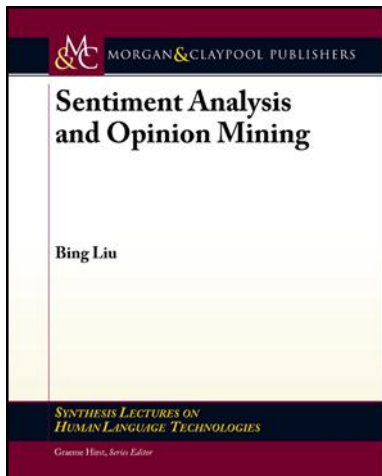


# Sentiment Analysis and Opinion Mining

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

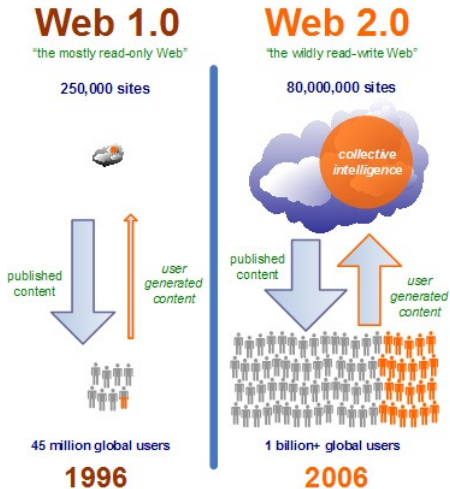


# The Task



- 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



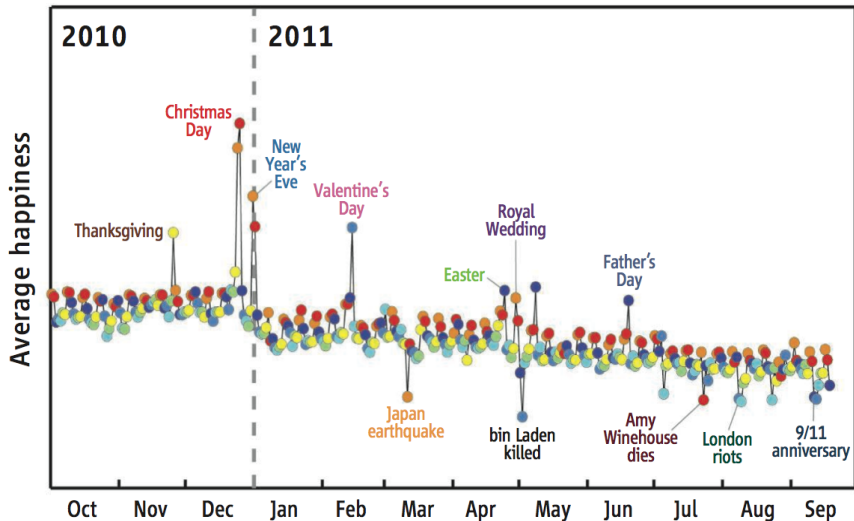
# Opinion Research / Market Research via Surveys




- 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



**HOLLYWOOD STOCK EXCHANGE**  
THE ENTERTAINMENT MARKET

Sign Up | Login  
Trade Movies, Stars & More!

MY PORTFOLIO | ON THE EXCHANGE | NEWS & EVENTS | COMMUNITY | EARN HS |  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
<b>After Earth</b> (Sony)	<a href="#">\$37.0M</a>	\$36.0M	\$33.0M	\$96.0M
<b>Now You See Me</b> (Summit)	<a href="#">\$24.0M</a>	\$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
<b>Fast &amp; Furious 6</b> (Universal)	2	\$130.8M	\$38.0M (-61%)	<a href="#">\$253.0M</a>
<b>The Hangover Part III</b> (Warner Bros.)	2	\$69.4M	\$18.8M (-55%)	<a href="#">\$115.0M</a>
<b>Epic</b> (Fox)	2	\$47.0M	\$20.1M (-45%)	<a href="#">\$95.0M</a>
<b>Star Trek Into Darkness</b> (Paramount)	3	\$162.1M	\$20.5M (-45%)	<a href="#">\$198.0M</a>
<b>The Great Gatsby</b> (Warner Bros.)	4	\$120.8M	\$7.3M (-46%)	<a href="#">\$134.0M</a>

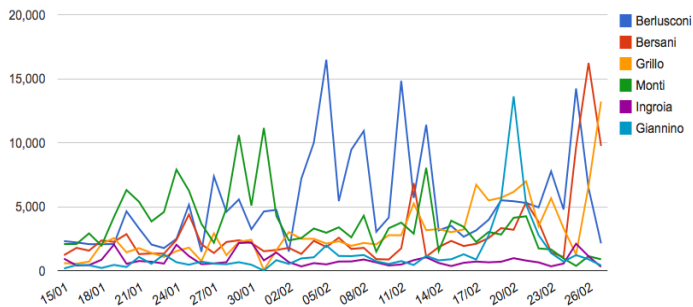
Forecasts and grosses as of May 30. Click on the hyperlinked numbers above to see the latest HSX forecasts.

App-driven live trading.  
What could be better than that?



# Political Science: Predicting Election Results

Confronto tra i candidati: Tutte le menzioni | Menzioni positive | Menzioni negative



# Online Reputation Detection / Management

**RestoreReputations.com**  
Reputation Management & Page Removal Service

## Online Reputation Management

Need defense against negative PR? Let us protect your image and increase positive perception.

More Info »

CALL: +63 918 550 9889

Reputation Management  
**Starts at \$99**  
a month

HOME ABOUT US CLEAN YOUR REPUTATION! SEO TACTICS **STATISTICS** SOLUTIONS SERVICES CONTACT US

### Online Reputation Confidential Online Reputation Management

#### Online Reputation Repair & Problem Prevention

### Reputation Management Services

Many companies experience challenges from hostile customers and even from their competitors who try to hurt the business which can directly affect the name, brand and reputation of the company by posting or blogging negative statements online about the company or any individual. They basically aim to adversely impact a target's business, recruiting and retention efforts, and the company's reputation as a whole. Search engine optimization (SEO) tool is a technology that reports search engines visibility details and monitors the health of the website. Thus search engine reputation management service is a way through which a client or a company can protect their fame, brand or reputation against negative, inaccurate and false publicity. And our search reputation management strategy is to replace the negative listings with the positive and favorable ones which you can control or influence.

**Click to Request a Quote**

# Computational Advertising

Omaha World Herald  73°

# Omaha.com

Search

Narrow search to » All | News | Sports | Money | Living | Go | More | Archive

Latest News Popular

- High court hears Kofoed appeal
- Breaking Brad: Friday, Sept. 9
- Iowa rapist gets 140 years in Neb.
- River could drop below flood stage
- Gov. critical of audit's release
- Omaha fire appeal dismissed
- Helicopter blade slashes man's face
- Small fire sparks Burke evacuation
- 1 critically injured in accident
- Omaha gets some Lincoln mail
- Iowa students test better
- NU may ease path to 4-year degrees
- Remender treats daughter
- **failblog.org** but tweaked
- "Road to nowhere" to be a memory

Autos Homes Rentals Jobs Classifieds Legalz Odds Find a Business Ads Baccos

HOME NEWS SPORTS MONEY LIVING ENTERTAINMENT LIVE WELL NEBRASKA

Featured: OVIHistory Premium Content Sex & Relationships Prep Zone

# FAIL

Published Wednesday September 7, 2011

## Nebraskan dies in ATV accident

« MetroRegion

 News Alerts

 Like

 Share  

CENTER, Neb. (AP) — A 50-year-old northeast Nebraska man has been killed in an accident on his all-terrain vehicle.

The Knox County Sheriff's Office said the body of Kelly Kracht, of Center, and his ATV were found in a creek bed around 6:40 p.m. Sunday.


Kracht had told his father that he was going to a pasture about one mile west of Center to treat a sick calf.

When he didn't return, a search was organized.

Investigators said it appeared that Kracht was on his ATV, chasing a calf, when the ATV went over the creek bank edge and rolled 18 feet to the bottom.

He was pronounced dead at the scene.



 GO FOR A RIDE

# 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
- 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  
→ “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 become 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

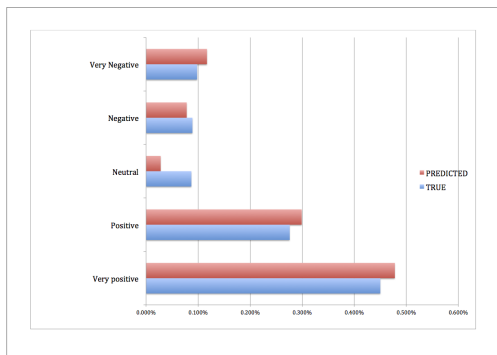


Aspect	Sentiment Score
money, price, cost, ...	★★★★★
ram, memory, ...	★★★
design, color, feeling, ...	★★★★★
extras, keyboard, screen, ...	★★

- 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**<sup>10</sup>
  - E.g., tweets, product reviews



<sup>10</sup>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.

Questions?



Thank you!

For any question, Skype me at fabseb60

