Faculty of Computer Science, Dalhousie University *28-Sep-2023*

CSCI 4152/6509 — Natural Language Processing

Lecture 8: Text Mining Review

Location: Rowe 1011 Instructor: Vlado Keselj
Time: 16:05 – 17:25

Previous Lecture

- P0 reminder: due this Friday by midnight, submission by email
  - Some possible sources of data and ideas for projects: Kaggle, TREC, PAN workshops, etc.
  - Guest speaker next lecture
- N-grams definition
- Extracting and Analyzing n-grams in Perl
- Elements of Information Retrieval
- Vector space model

After preprocessing steps, such as stop-word removal, rare words removal, and stemming, we have a global set of terms $\{t_1, t_2, \ldots, t_m\}$, which are used to represent documents and queries.

In a vector space model, document and queries are represented by vectors of weights, such as

$$\vec{d} = (w_{1,d}, w_{2,d}, \ldots, w_{m,d}), \text{ and } \vec{q} = (w_{1,q}, w_{2,q}, \ldots, w_{m,q})$$

where the weights $w_{i,x}$ correspond to the term t_i, of the document or query x. There are different ways how weights can be determined. One simple way is to use *binary* weights: 1 if the document contains the term, or 0 if it does not. Another option is to use term counts, or frequency within the document or query. The most widely adopted standard choice is to use *term frequency inverse document frequency* weights (*tfidf*), which are calculated using the following formula:

$$\mathit{tfidf} = \mathit{tf} \cdot \log\left(\frac{N}{\mathit{df}}\right)$$

where *tf* is frequency (count) of a term in document, which is sometimes log-ed as well; *df* is document frequency, i.e., number of documents in the collection containing the term; and N is the total number of documents in the collection. The document frequency df is the number of documents that contain the term t. We could also calculate it as the portion of the document collection that contain the term; i.e., the fraction of documents that contain the term, which would be df/N. A term should be more important and have a higher weight if it is more rare, so that is the reason why we use the inverse document frequency, or N/df. For very rare terms this number could be very large, for example the terms with $\mathit{df} = 1$ and $N = 1\,000\,000$ it would be $1\,000\,000$, so to “curb” this growth we apply the slow-growing logarithm function and finally obtain $\mathit{tfidf} = \mathit{tf} \cdot \log(N/\mathit{df})$. In some references, the logarithm is applied to tf as well.

8.3 Cosine Similarity Measure

A natural measure to measure similarity between a document and a query is the *cosine similarity measure.* It is known that the cosine of the angle between two vectors can be easily computed using the following formula:

$$\mathit{sim}(q,d) = \frac{\sum_{i=1}^{m} w_{i,q} w_{i,d}}{\sqrt{\sum_{i=1}^{m} w_{i,q}^2} \cdot \sqrt{\sum_{i=1}^{m} w_{i,d}^2}} = \frac{\vec{q} \cdot \vec{d}}{|\vec{q}| \cdot |\vec{d}|}$$

The formula gives the cosine of the angle between vectors in 2-dimensional and 3-dimensional space, and although we cannot exactly image the angle in spaces in more dimensions it still preserve some nice properties that match our intuition about similarity between documents, or between a document and a query. For example, if a document and query have exactly the same terms and in exactly the same proportion of their frequencies, the angle will be 0, and the cosine will be 1. On the other hand, if and only if the query and the document have no terms in common, the angle will be $90°$, and the cosine will be 0.

The angle between a query and a document vector in the 3-dimensional space is shown in the Figure 1.

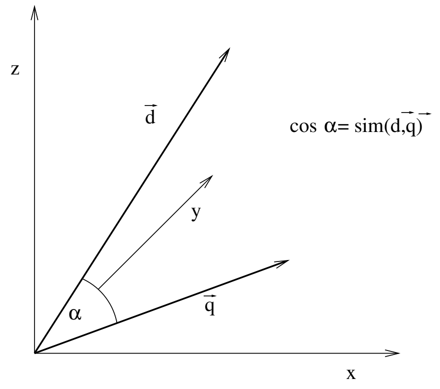


Figure 1: Cosine Similarity in the 3-dimensional Space

If the vectors representing documents ($\vec{d}$) and queries ($\vec{q}$) are normalized in advance, i.e., if they are divided with their length, then the cosine similarity computation becomes simpler and more efficient. Namely, if the normalized vectors are precomputed

$$\vec{d_0} = \frac{\vec{d}}{|\vec{d}|} = \left(\frac{w_{1,d}}{\sqrt{\sum_{i=1}^m w_{i,d}^2}}, \frac{w_{2,d}}{\sqrt{\sum_{i=1}^m w_{i,d}^2}}, \dots \frac{w_{m,d}}{\sqrt{\sum_{i=1}^m w_{i,d}^2}} \right)$$

and

$$\vec{q_0} = \frac{\vec{q}}{|\vec{q}|} = \left(\frac{w_{1,q}}{\sqrt{\sum_{i=1}^m w_{i,q}^2}}, \frac{w_{2,q}}{\sqrt{\sum_{i=1}^m w_{i,q}^2}}, \dots \frac{w_{m,q}}{\sqrt{\sum_{i=1}^m w_{i,q}^2}} \right)$$

then the similarity value is simply computed as

$$\mathit{sim}(q,d) = \vec{q_0} \cdot \vec{d_0} = \sum_{i=1}^{m} w_{iq_0} w_{id_0}$$

Side Note: An interesting open-source search engine:

- Lucene search engine
- `http://lucene.apache.org`
- Open-source, written in Java
- Uses the vector space model
- Another interesting link: Introduction to IR on-line book covers well text classification:
  `http://nlp.stanford.edu/IR-book/html/htmledition/irbook.html`

8.4 Term-by-Document Matrix and Latent Semantic Analysis

Note: This subsection will not be covered in the class.

Term-by-Document Matrix: The vector space model provides a way to represent each document as a vector. If we have m selected terms for a document collection of n documents, then using for example *tfidf* weights we can represent each of the n documents as an m-dimensional vector. If we order these vectors as columns, we get an $m \times n$ dimensional matrix called *term-by-document matrix.* if w_{ij} is the weight of term t_i in the document d_j, then the term-by-document matrix is $[w_{ij}]_{m\times n}$, or:

$$\begin{array}{c|cccc} & d_1 & d_2 & \dots & d_n \\ \hline t_1 & w_{11} & w_{12} & \dots & w_{1n} \\ t_2 & w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ t_m & w_{m1} & w_{m2} & \dots & w_{mn} \end{array} \quad \text{or, as a proper matrix:} \quad \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ w_{m1} & w_{m2} & \dots & w_{mn} \end{bmatrix}$$

Dimensionality reduction: The number of different terms generally corresponds to the number of different words in a document collection, and this number is generally large, in the range of 100,000s. It is useful for various application to reduce this dimensionality. Some ways of reducing dimensionality are: removing stop-words and very rare words, and selecting only the most distinctive terms, which is a process known as feature selection.

Latent Semantic Analysis: An interesting mathematical way of representing documents in a vector space with much lower dimensionality is known as Latent Semantic Analysis.

Latent Semantic Analysis

- A method for term-by-document dimensionality reduction
- Also known as Latent Semantic Indexing (LSI) in IR
- Example with four terms and two documents
- Main idea: use singular value decomposition on term-by-document matrix M
- Singular value decomposition: $M_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^T_{n\times n}$
- Closest by Frobenius norm matrix of rank $\le k$ is
  $M^{(k)}_{m\times n} = U_{m\times m}\Sigma^{(k)}_{m\times n}V^T_{n\times n}$
- Concept and document representations

8.5 IR Evaluation Measures: Precision, Recall, and F-measure

Note: This section is normally covered in an earlier Machine Learning course, so it is covered in the class as a review material.

We will now define some main evaluation measures used in IR, which are also important in general text mining tasks, such as text classification. The main three measures are: precision, recall, and F-measure:

- **Precision** is the percentage of true positives out of all returned documents; i.e.,

$$P = \frac{\mathit{TP}}{\mathit{TP} + \mathit{FP}}$$

- **Recall** is the percentage of true positives out of all relevant documents in the collection; i.e.,

$$R = \frac{\mathit{TP}}{\mathit{TP} + \mathit{FN}}$$

- **F-measure** is a weighted harmonic mean between Precision and Recall:

$$F = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

– We usually set $\beta = 1$, in which case we have:

$$F = \frac{2PR}{P+R}$$

Precision and Recall can be explained in the following way: Given a query, the search engine identifies a set of relevant documents, and returns this set. Some of the returned documents are truly relevant and we call them *true positives (TP);* some returned documents are not relevant and we call them *false positives (FP);* some documents are relevant but were not returned by the engine and we call them *false negatives (FN);* and the remaining documents are not returned by the engine and they are not relevant, and we call them *true negatives (TN).*

The typical value in the F-measure is $\beta = 1$, for equal emphasis on precision and recall. However, if we want to put more emphasis on precision we choose β close to 0, and for more emphasis on recall we choose β close to 1. β must always be from the interval $[0, 1]$, i.e., $0 \leq \beta \leq 1$.

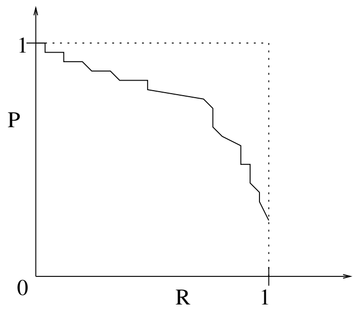
8.6 Recall-Precision Curve

Note: Covered in class as review material.

Precision, recall and F-measure may be seen as too simplistic views of evaluating a search engine since a search engine does not return just a set of relevant documents, but a ranked list of relevant documents.

A more appropriate way to evaluate this ranked list is using the *recall-precision curve.* The basic idea of the recall-precision curve is to draw a point in a two-dimensional plane corresponding to precision (y axis) and recall (x axis) of the following document sets: first ranked document, the first two ranked documents, the first three ranked documents, and so on. Although such curve is a roughly smooth curve going from the point (0,1) to (1,0), it does usually have some noisy changes of direction, so it is "smoothed" by actually using the interpolated precision curve.

A typical Recall-Precision curve looks as follows:



To avoid "noisy" changes of curve direction, *interpolated precision* (*IntPrec*) is often used. Interpolated precision is the maximal precision that is obtained with certain recall level; namely, if $P(k)$ and $R(k)$ are precision and recall of the set of first k ranked documents, than for any recall value $r \in [0, 1]$:

$$\mathit{IntPrec}(r) = \max_{k, R(k) \geq r} P(k)$$

Recall-Precision Curve Example. Suppose that a search engine returned 12 ranked results to our query, and when we checked them, the following are our judgments on their relevance:

1. relevant
2. relevant
3. relevant
4. not relevant
5. relevant
6. not relevant

7. relevant
8. not relevant
9. not relevant
10. relevant
11. not relevant
12. not relevant

Task 1: Precision, Recall and F-measure

– Assuming that the total number of relevant documents in the collection is 8, calculate precision, recall, and F-measure ($\beta = 1$) for the returned 12 results.

Since there is a total of 6 relevant documents among the set of 12, we can calculate precision to be $P = \frac{6}{12} = 0.5$.

It is assumed that there is a total number of 8 relevant documents in the collection, so the recall is $R = \frac{6}{8} = 0.75$.

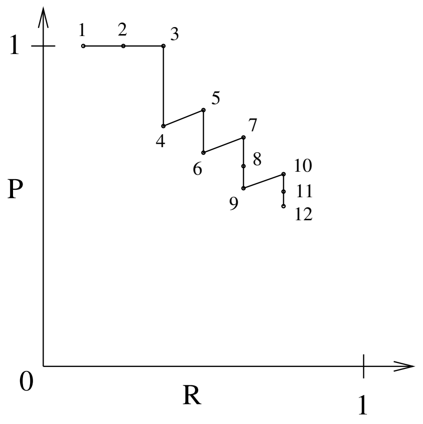
Finally, we calculate the F-measure: $F = \frac{2PR}{P+R} = \frac{2 \cdot 0.5 \cdot 0.75}{0.5+0.75} = \frac{0.75}{1.25} = \frac{3}{5} = 0.6$.

Task 2: Recall-Precision Curve

– Task: Draw the recall-precision curve for these results
– First step: Form sets of n initial documents, and look at their relevance:

Set 1: $\{R\}$:	$R_1 = \frac{1}{8} = 0.125 \quad P_1 = \frac{1}{1} = 1$
Set 2: $\{R, R\}$:	$R_2 = \frac{2}{8} = 0.25 \quad P_2 = \frac{2}{2} = 1$
Set 3: $\{R, R, R\}$:	$R_3 = \frac{3}{8} = 0.375 \quad P_3 = \frac{3}{3} = 1$
Set 4: $\{R, R, R, NR\}$:	$R_4 = \frac{3}{8} = 0.375 \quad P_4 = \frac{3}{4} = 0.75$
Set 5: $\{R, R, R, NR, R\}$:	$R_5 = \frac{4}{8} = 0.5 \quad P_5 = \frac{4}{5} = 0.8$
Set 6: $\{R, R, R, NR, R, NR\}$:	$R_6 = \frac{4}{8} = 0.5 \quad P_6 = \frac{4}{6} \approx 0.666666666666667$
Set 7: $\{R, R, R, NR, R, NR, R\}$:	$R_7 = \frac{5}{8} = 0.625 \quad P_7 = \frac{5}{7} \approx 0.714285714285714$
Set 8: $\{R, R, R, NR, R, NR, R, NR\}$:	$R_8 = \frac{5}{8} = 0.625 \quad P_8 = \frac{5}{8} = 0.625$
Set 9: $\{R, R, R, NR, R, NR, R, NR, NR\}$:	$R_9 = \frac{5}{8} = 0.625 \quad P_9 = \frac{5}{9} \approx 0.555555555555556$
Set 10: $\{R, R, R, NR, R, NR, R, NR, NR, R\}$:	$R_{10} = \frac{6}{8} = 0.75 \quad P_{10} = \frac{6}{10} = 0.6$
Set 11: $\{R, R, R, NR, R, NR, R, NR, NR, R, NR\}$:	$R_{11} = \frac{6}{8} = 0.75 \quad P_{11} = \frac{6}{11} \approx 0.545454545454545$
Set 12: $\{R, R, R, NR, R, NR, R, NR, NR, R, NR, NR\}$:	$R_{12} = \frac{6}{8} = 0.75 \quad P_{12} = \frac{6}{12} = 0.5$

Using these ten points, we can draw the recall-precision curve:



The recall-precision curve that we just saw is not exactly monotonically non-increasing although the general trend of these curves is to grom from near the point $(0, 1)$ down towards point $(1, 0)$ in the coordinate system. The reason is that after the initial 100% precision, each time we see a new relevant document in the ranked list both precision and recall will increase a bit, and when we see a non-relevant document the precision will drop with the same recall, which creates a bit of a zig-zagged line. To make the non-increasing, and thus a bit smoother, we use the *interpolated precision-recall curve,* which is obtained by keeping precision at the level that is maximal from the associated recall point and forward.

Slide notes:

Task 3: Interpolated Recall-Precision Curve

- Task: Draw interpolated Recall-Precision curve
- Formula:
$$\textit{IntPrec}(r) = \max_{k, R(k) \geq r} P(k)$$
- Based on the previous Task:
$0 \leq r \leq R_4 = \frac{3}{8} = 0.375 \Rightarrow \textit{IntPrec}(r) = 1$
$R_4 < r \leq R_6 = \frac{4}{8} = 0.5 \Rightarrow \textit{IntPrec}(r) = 0.8$
$R_6 < r \leq R_9 = \frac{5}{8} = 0.625 \Rightarrow \textit{IntPrec}(r) = 5/7 \approx 0.714285714$
$R_9 < r \leq R_{12} = \frac{6}{8} = 0.75 \Rightarrow \textit{IntPrec}(r) = 0.6$

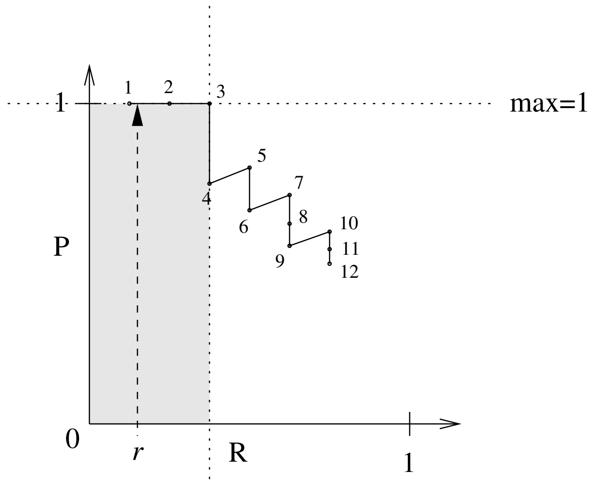
To calculate the interpolated recall-precision curve, we use the formula:

$$\textit{IntPrec}(r) = \max_{x, R(k) \geq r} P(k)$$

To use this formula, we can start first with $r = 0$, which gives:

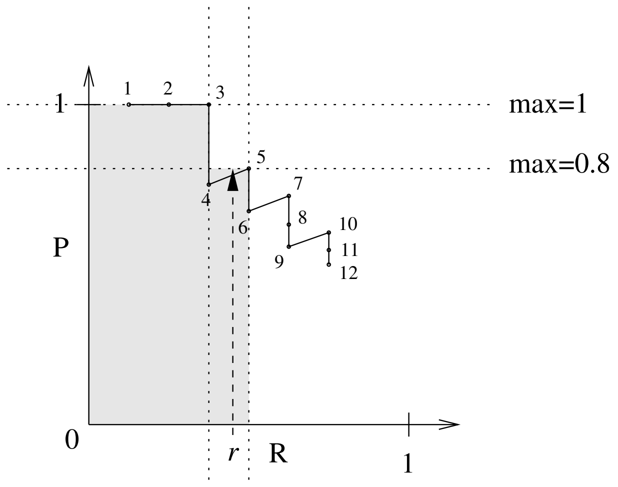
$$\textit{IntPrec}(0) = \max_{k, R(k) \geq 0} P(k) = \max_{k} P(k) = 1$$

because $R(k) \geq 0$ for all k, and maximum $P(k)$ is 1. If we increase r starting from 0, we see that the maximal precision will remain 1 for all points R_1, R_2, R_3, and R_4, as shown in the following figure:



This is how we get the following values for the Interpolated Precision:
$0 \leq r \leq R_4 = \frac{3}{8} = 0.375 \Rightarrow \textit{IntPrec}(r) = 1$

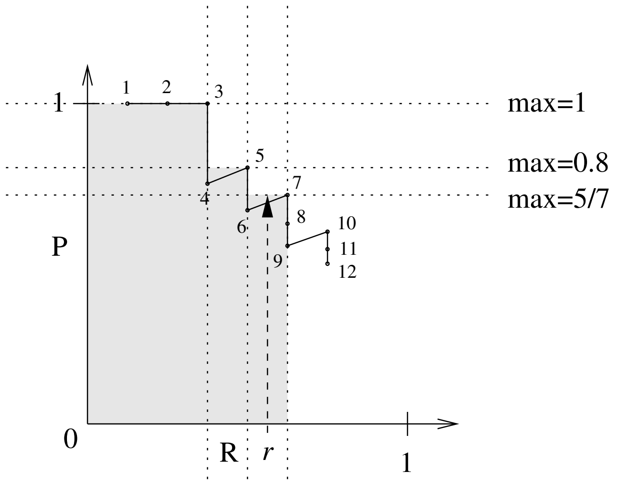
For the values $r > R_4$, the next maximum is 0.8, as shown in the following figure:



and that is how we obtain the next interval:

$R_4 < r \leq R_6 = \frac{4}{8} = 0.5 \Rightarrow \textit{IntPrec}(r) = 0.8$

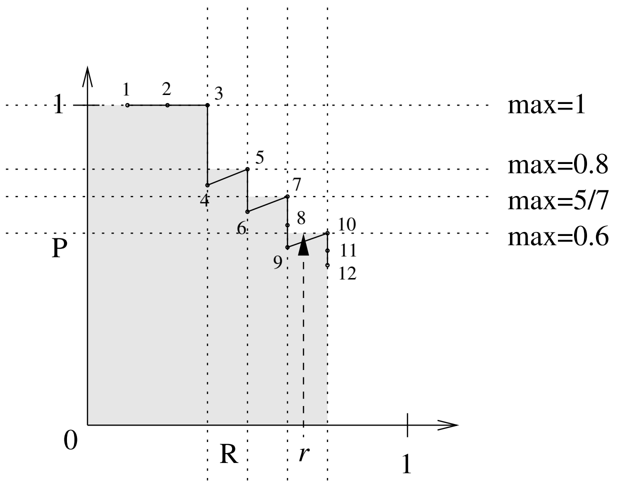
Similarly, for the values $r > R_6$, the next maximum is $5/7$, as shown in the following figure:



and that is how we obtain the next interval:

$R_6 < r \leq R_9 = \frac{5}{8} = 0.625 \Rightarrow \textit{IntPrec}(r) = 5/7 \approx 0.714285714$

For the values $r > R_9$, the next maximum is 0.6, as shown in the following figure:

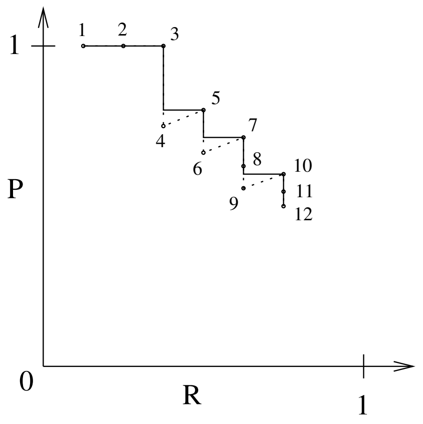


and that is how we obtain the final interval:

$R_9 < r \leq R_{12} = \frac{6}{8} = 0.75 \Rightarrow \textit{IntPrec}(r) = 0.6$

There are no further points, so we can finish the curve at this point. We can summarize the values for the interpolated recall-precision curve as:
$0 \leq r \leq R_4 = \frac{3}{8} = 0.375 \Rightarrow \textit{IntPrec}(r) = 1$
$R_4 < r \leq R_6 = \frac{4}{8} = 0.5 \Rightarrow \textit{IntPrec}(r) = 0.8$
$R_6 < r \leq R_9 = \frac{5}{8} = 0.625 \Rightarrow \textit{IntPrec}(r) = 5/7 \approx 0.714285714$
$R_9 < r \leq R_{12} = \frac{6}{8} = 0.75 \Rightarrow \textit{IntPrec}(r) = 0.6$

Using these points, we can construct the interpolated precision curve, up to $R = 0.75$:



A way to look at the interpolated recall-precision curve is that it is obtained from the direct recall-precision curve by "filling up" all parts where the curve increases, in a way similar as water would fill up the "depressions" if we imagine the curve to be a waterfall cascade.

Interpolated R-P Curve at 11 Standard Levels

Some Other Similar Measures

– Fallout

$$\mathit{Fallout} = \frac{FP}{FP+TN}$$

– Specificity

$$\mathit{Specificity} = \frac{TN}{TN+FP}$$

– Sensitivity

$$\mathit{Sensitivity} = \frac{TP}{TP+FN} \quad (= R)$$

– Sensitivity and Specificity: useful in classification and contexts such as medical tests

Sensitivity and Specificity are typically used in the classification task, that we will described later. They are also frequently useful to evaluate medical tests. For example, if we consider a context of a medical test for disease diagnostics: a sensitive test, same as recall, is good at not missing any true cases of a disease (true positives); while a specific test is good in eliminating a possible suspected disease.

9 Text Classification as General NLP Task

Note: Covered in class as review material.

9.1 Text Classification as a Text Mining Task

Text Classification is one of the tasks in a more general area called *Text Mining*. The area of Text Mining generally deals with processing of large quantities of text and deriving some useful information, knowledge, or insight from it. The name Text Mining is derived from a similar area of Data Mining, which deals with large quantities of data in general for the purpose of knowledge discovery. Some tasks in Text Mining are related to similar tasks in a wider Data Mining area. The following are some typical text mining tasks:

- Text Classification
- Text Clustering
- Information Extraction
- And some new and less prominent tasks:
  - Text Visualization
  - Filtering tasks, Event Detection
  - Terminology Extraction

Text Classification is the task of classifying documents into classes of documents; i.e., sets of documents, of certain properties. For example, classifying email into spam email and non-spam email is an example of text classification. *Text Clustering* is the task of grouping documents in a collection in a groups of similar documents, or clusters. *Information Extraction* is the task of extracting table-like data from text documents. For example, processing news and filling out information about companies, their names, addresses, and names of CEOs would be an information extraction task.

The area of text visualization addresses the problem of different visual representations of text and textual documents. Filtering tasks deal with selecting relevant documents or information from a stream of usually short textual documents. Event detection is the task of detecting events from a stream of documents, such as news-wire. A CEO change in a company could be one kind of interesting events to be detected. Terminology Extraction is the task of extracting terminology, i.e., domain terms, from a document collection in a domain. For example, analyzing bio-medical scientific papers and detecting and extracting terminology such as protein names would be an example of terminology extraction.

9.2 Types of Text Classification

Text Classification is also known as Text Categorization. It is the problem of automatically classifying a document into one of predetermined classes or categories of documents. In a more usual form of classification, we always assign a document to exactly one class. In a more flexible form, known as *multi-label* classification, we assign document to zero or more classes; or we can view the task as assigning a set of labels to the document, where each label is a designation of a class.

Beside some reading on document classification in the textbook [JM], there is a more elaborate description in the Manning and Schütze book ([MS]), in Chapter 16: Text Categorization.

Types of Text Classification

- topic categorization
- sentiment classification
- authorship attribution and plagiarism detection
- authorship profiling (e.g., age and gender detection)
- spam detection and e-mail classification
- encoding and language identification
- automatic essay grading

More specialized example: dementia detection using spontaneous speech

Creating Text Classifiers

- Can be created manually
  - typically rule-based classifier
  - example: detect or count occurrences of some words, phrases, or strings
- Another approach: make programs that *learn* to classify
  - In other words, classifiers are generated based on labeled data

– supervised learning

While we can create a classifier from the scratch, a more usual approach is to classify, i.e., label, a set of document manually, and then devise a method generate a classifier based on this set of labeled documents. This process of generating a classifier is known as *training*, or *machine learning*. The classification problem is known as an example of *supervised learning* in the machine learning area, since we need to provided labeled examples; in other words, we "supervise" the learning algorithm.

9.3 Evaluation Measures for Text Classification

When we build a text classifier, an important question is how to evaluate it so we can measure how good it is and how different classifiers compare. In order to do this, we first need to prepare a dataset consisting of documents, where each document is labeled with the class that it belongs too. This is called a *labeled dataset* or a *labeled document collection.* The labels are usually assigned and checked manually, and we take them as the *ground truth,* also called the *gold standard,* against which we evaluate the classifier.

We run the classifier against the documents, with labels being hidden, and once the classifier assigns labels to the documents, we compare them against the gold standard labels. The first evaluation measure that we can calculate is the *accuracy,* which is the percentage or fraction of the documents being correctly classified by the classifier. If we want to examine in more details how classifier performed on documents from different classes, we present the results in the *contingency table* also known as the *confusion matrix.*

Slide notes:

Evaluation Measures for Text Classification

- Contingency table (confusion matrix) and Accuracy
- Example (classes A, B, and C):

		Gold standard			
		A	B	C	
Model classification	A	5	1	1	7
	B	3	10	2	15
	C	0	2	10	12
		8	13	13	34

- Accuracy: percentage of correct classifications; in the example, $= 25/34 \approx 0.7353 = 73.53\%$

Per class: Precision, Recall, and F-measure

- For each class: Yes = in class, No = not in class

	Yes is correct	No is correct
Yes assigned	a	b
No assigned	c	d

- precision ($\frac{a}{a+b}$), recall ($\frac{a}{a+c}$), fallout ($\frac{b}{b+d}$), F-measure:

$$F = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

- If $\beta = 1 \Rightarrow$ Precision and Recall treated equally
- macro-averaging (equal weight to each class) and micro-averaging (equal weight to each object) (2×2 contingency tables vs. one large contingency table)

Example: Let us assume that we are evaluating an authorship-attribution classifier. The classifier is presented with 34 documents, written by three authors: A1, A2, and A3, and the classifier produces labels. We know the true authorship of the documents, so we compare the classifier results to these labels. The obtained results can be presented as a so-called *confusion matrix*, or *contingency table*:

		Gold standard			
		A1	A2	A3	
System response	A1	5	1	1	7
	A2	3	10	2	15
	A3	0	2	10	12
		8	13	13	34

Or, we can create contingency tables for each class separately:

	Gold standard		
	A1	not A1	
A1	5	2	7
not A1	3	24	27
	8	26	34

	Gold standard		
	A2	not A2	
A2	10	5	15
not A2	3	16	19
	13	21	34

	Gold standard		
	A3	not A3	
A3	10	2	12
not A3	3	19	22
	13	21	34

The overall accuracy can be calculated using the overall table;

$$Accuracy = \frac{5+10+10}{34}$$

Per-class precisions are:

$$P_{A1} = \frac{5}{7} \quad P_{A2} = \frac{10}{15} \quad P_{A3} = \frac{10}{12}$$

Per-class recalls are:

$$R_{A1} = \frac{5}{8} \quad R_{A2} = \frac{10}{13} \quad R_{A3} = \frac{10}{13}$$

Macro-averaged precision, recall, and F-measure are:

$$P_{macro} = \frac{5/7 + 10/15 + 10/12}{3} \quad R_{macro} = \frac{5/8 + 10/13 + 10/13}{3} \quad F_{macro} = \frac{2 \cdot P_{macro} \cdot R_{macro}}{P_{macro} + R_{macro}}$$

To calculate micro-averaged precision, recall, and F-measure, we calculate cumulative per-class table:

	Gold standard		
	A	not A	
A	25	9	34
not A	9	59	68
	34	68	102

and then we calculate the micro-averaged measures:

$$P_{micro} = \frac{25}{34} \quad R_{micro} = \frac{25}{34} \quad F_{micro} = \frac{2 \cdot P_{micro} \cdot R_{micro}}{P_{micro} + R_{micro}} = \frac{25}{34}$$

9.4 Evaluating Classifiers

In order to decide which machine learning classification method is the best choice for a classification task, we need to evaluate those methods, or classifiers for short. The goal of evaluation is to see what kind of performance we can expect from a classifier after it learns its model from training data. The two main concerns with classifier learning are underfitting and overfitting.

9.4.1 Underfitting and Overfitting

Slide notes:

Evaluation Methods for Classification

- General issues in classification
  - Underfitting and Overfitting
- Example with polynomial-based function learning
  - Underfitting and Overfitting

Two basic issues that we can find with classifiers that are trained from labeled data are *underfitting* and *overfitting*. *Underfitting* is an issue in which the training algorithm does not learn well from training data, which is indicated by poor performance on the training data itself. This means that it cannot *fit* well the training data. In this case, we cannot expect any better performance on unseen data. *Overfitting* is the issue that the classifiers fits very well the training data, and even too well in sense that it learns signals from training data that are not relevant to the task, which hurt its performance on unseen data. In other words, the classifier fits the training data too well, and thus it does not properly generalize in order to classify unseen data. These issues can be well illustrated on an example of polynomial-based function learning.

Polynomial-based Function Learning Example. An example for overfitting and underfitting can be created from function learning. We can draw a set of, for example, 10 points in a 2-dimensional plane, being closely distributed around a parabola with equation $y = x^2/4$. Assuming that we do not know that the points are generally close to this parabola, we could try to fit them to polynomials of various degrees. By fitting them with a polynomial with the first degree, i.e., a line, we can see that the data does not fit the line very well. We call this a training error, since our line is learned from the 10 training points, and this is an example of underfitting, since we cannot fit the training data. On the other hand, if we try a polynomial of 9th degree, we can perfectly fit the training data, but the polynomial will not match well new data. This is an overfitting error, since the learned function fits the training data too well, fitting even the errors which may have happened in measurements or similar.

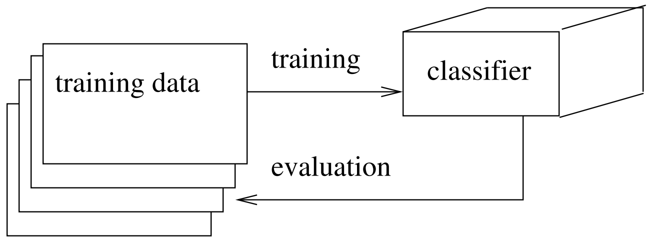
9.4.2 Evaluation Methods for Text Classifiers

Slide notes:

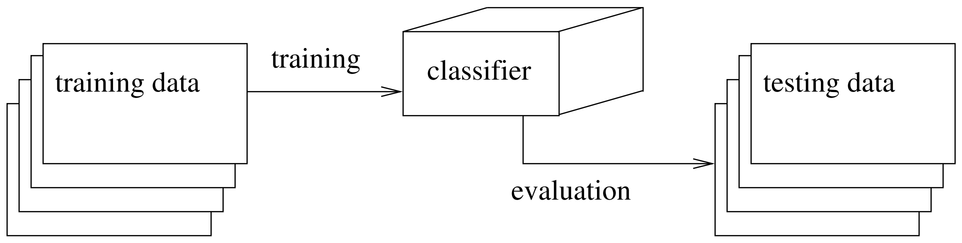
Evaluation Methods for Text Classifiers

- Training Error
- Train and Test
- N-fold Cross-validation

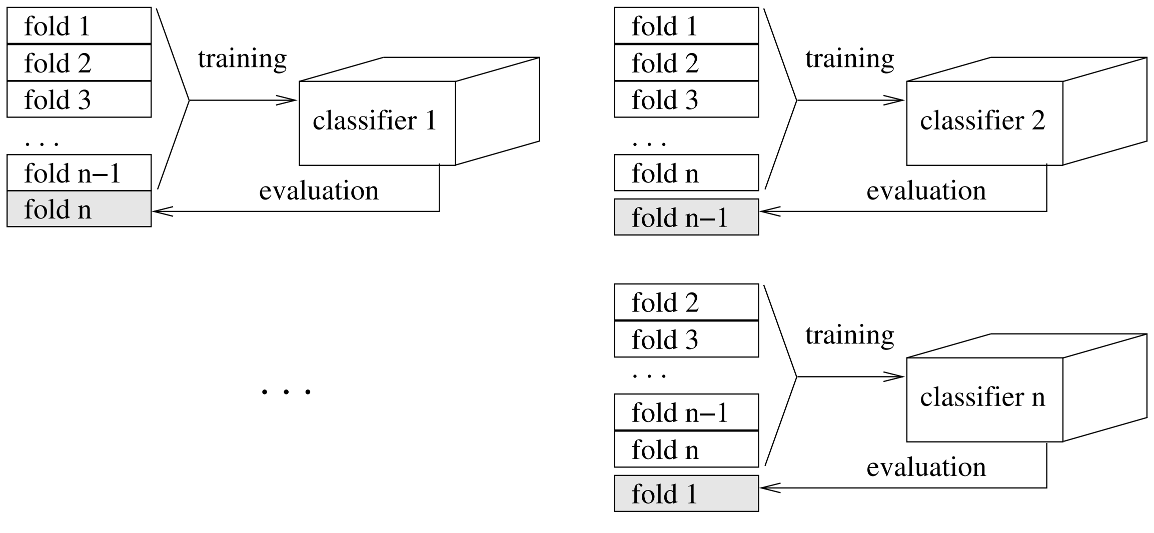
Training Error. The classifier is trained on a training data set and also evaluated on the same data set. It is a good idea to get this result, although it is obviously *biased* towards the training data. This evaluation can detect underfitting but not overfitting of the training data.



Train and test. The data is divided into two parts: training and testing part. The split is usually 90% for training and 10% for testing, but sometimes 2/3 of data is used for training and 1/3 for testing. This is an unbiased evaluation, which can detect underfitting as well as overfitting. To be sure that the evaluation is unbiased, it is important not to use testing data in any way, even to glance at it, if it may influence our decisions regarding classifier construction. With some methodologically generic methods, this is not an issue.



N-fold cross-validation. In this method, the data is randomly partitioned into n equal parts. n experiments are performed, where in each another part is taken as the testing data, while remaining parts are used for training. At the end, the results over experiments are averaged. This is unbiased testing that gives more statistical significance than train-and-test, but it is not applicable if we need to examine the training data during classifier construction.



Side note: An interesting link:

– SpamAssassin is considered the best, or at least one of the best spam-classification software packages. It is an open-source package available at:
`http://spamassassin.apache.org`

9.5 Text Clustering

Note: *This sub-subsection (Text Clustering) is not covered in this course. These notes are only for your additional information, or use in a project.*

- Task definition
- Example of unsupervised learning
- Example approach: the simple k-means approach
- Hierarchical clustering
  - agglomerative, and
  - divisive
- Evaluation
  - inter-cluster similarity (average inter-cluster distance)
  - cluster purity (classes known)
  - entropy or information gain (classes known)

Simple k-means. In the simple k-means clustering, all documents are translated into vectors of same dimensionality. We choose a number of clusters k in advance and choose randomly initial k centroids, i.e., vectors of the same dimensionality as the documents. After this we repeat the following two steps:

1) Assign each document to the closest centroid using Euclidean distance. In this way, all documents are partitioned into k clusters.

2) Calculate new k centroids as centroids of k clusters created in the previous step. A centroid of a set of vectors is equivalent to the arithmetic mean of a set of numbers, i.e., for n vectors v_1, v_2, …, v_n, their centroid is calculated using the formula:

$$\textit{centroid} = \frac{\sum_{i=1}^{n} v_i}{n}$$

This iterative process stops either if: (1) the clusters do not change between two iterations, i.e., a stable partition is achieved; (2) a clustering quality measure stops improving; or, (3) we reach a predefined iteration limit.

Clustering evaluation. The clustering evaluation methods can be *internal* and *external.* In the internal methods, clusters are evaluated based on the high similarity among items in the same cluster, and low similarity among items in different clusters. In the external methods, clustering is evaluated as part of a larger application, and the performance of that application is measured.

Average inter-cluster distance. As an example for internal evaluation, we could find the average distance among all pairs of items belonging to the same cluster. This is the average inter-cluster distance. The lower the average inter-cluster distance, the better is the clustering. The number of clusters needs to be limited from above in this case, since one-item-one-cluster approach would lead to the best (zero) average inter-cluster distance, with a very large number of clusters.

Clustering purity. The clustering purity measure can be used if we have document labels according to some classification. We assume that a better clustering will group more documents with the same label in one cluster, and tend to put documents with different labels in different clusters. Clustering purity evaluation can be described as follows: we assign each cluster its majority class and then we measure accuracy as it would be done for classification:

$$\text{purity}(\Omega, C) = \frac{1}{N} \Sigma_k \max_j |\omega_k \cup c_j|$$

where $\Omega = \{\omega_1, \ldots, \omega_K\}$ is the set of clusters, and $C = \{c_1, \ldots c_J\}$ is the set of classes.

There are other evaluation measures for clustering, including F-measure for clustering.