Text Classification via Supervised Learning: Techniques and Trends

Fabrizio Sebastiani
http://www.isti.cnr.it/People/F.Sebastiani/

Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
Via Giuseppe Moruzzi 1 – 56124 Pisa, Italy
E-mail: fabrizio.sebastiani@isti.cnr.it

Corso di Laurea Magistrale in Informatica
Information Retrieval
Anno Accademico 2010/11

My recent research ...

Outline

Outline

Defining TC

- Text classification (TC – aka text categorization) is the task of building text classifiers, i.e. software systems that classify documents from a domain $\mathcal{D}$ into a given, fixed set $\mathcal{C} = \{c_1, \ldots, c_m\}$ of pre-defined categories (aka classes, or labels).
- TC is an approximation task, in that we assume the existence of a "gold standard", or "ground truth" (target function) that specifies how documents ought to be classified (or: how a supposedly "expert" professional would classify them). Since this gold standard is unknown, the task consists in building a system that "approximates" it.
- Note that the term "classification" is sometimes used in the literature to also mean either
  - clustering, i.e. finding a yet undetected group structure on $\mathcal{D}$; or
  - "putting data items into groups", which subsumes both clustering and "categorization".

Defining TC

- Text classification (TC – aka text categorization) is the task of building text classifiers, i.e. software systems that classify documents from a domain $\mathcal{D}$ into a given, fixed set $\mathcal{C} = \{c_1, \ldots, c_m\}$ of pre-defined categories (aka classes, or labels).
- TC is an approximation task, in that we assume the existence of a "gold standard", or "ground truth" (target function) that specifies how documents ought to be classified (or: how a supposedly "expert" professional would classify them). Since this gold standard is unknown, the task consists in building a system that "approximates" it.
- Note that the term "classification" is sometimes used in the literature to also mean either
  - clustering, i.e. finding a yet undetected group structure on $\mathcal{D}$; or
  - "putting data items into groups", which subsumes both clustering and "categorization".

Defining TC

- Text classification (TC – aka text categorization) is the task of building text classifiers, i.e. software systems that classify documents from a domain $\mathcal{D}$ into a given, fixed set $\mathcal{C} = \{c_1, \ldots, c_m\}$ of pre-defined categories (aka classes, or labels).
- TC is an approximation task, in that we assume the existence of a "gold standard", or "ground truth" (target function) that specifies how documents ought to be classified (or: how a supposedly "expert" professional would classify them). Since this gold standard is unknown, the task consists in building a system that "approximates" it.
- Note that the term "classification" is sometimes used in the literature to also mean either
  - clustering, i.e. finding a yet undetected group structure on $\mathcal{D}$; or
  - "putting data items into groups", which subsumes both clustering and "categorization".

Defining TC (cont'd)

- For greater generality we will assume that the categories are just symbolic labels; in particular, we assume that (1) the "text" constituting the label is not significant, and that (2) no additional knowledge of the categories' "meaning" (e.g. lexical resources) is available.
- In an operational environment the assumptions above may not be verified. In this case we may use whatever source of knowledge might be available.
- The attribution of documents to categories should be realized on the basis of the content of the documents. Given that this is an inherently subjective notion, the membership of a document in a category (the fundamental notion of TC, akin to the IR notion of relevance of a document to an information need) cannot be determined with certainty.

Defining TC (cont'd)

- For greater generality we will assume that the categories are just symbolic labels; in particular, we assume that (1) the "text" constituting the label is not significant, and that (2) no additional knowledge of the categories' "meaning" (e.g. lexical resources) is available.
- In an operational environment the assumptions above may not be verified. In this case we may use whatever source of knowledge might be available.
- The attribution of documents to categories should be realized on the basis of the content of the documents. Given that this is an inherently subjective notion, the membership of a document in a category (the fundamental notion of TC, akin to the IR notion of relevance of a document to an information need) cannot be determined with certainty.

Defining TC (cont'd)

- For greater generality we will assume that the categories are just symbolic labels; in particular, we assume that (1) the "text" constituting the label is not significant, and that (2) no additional knowledge of the categories' "meaning" (e.g. lexical resources) is available.
- In an operational environment the assumptions above may not be verified. In this case we may use whatever source of knowledge might be available.
- The attribution of documents to categories should be realized on the basis of the content of the documents. Given that this is an inherently subjective notion, the membership of a document in a category (the fundamental notion of TC, akin to the IR notion of relevance of a document to an information need) cannot be determined with certainty.

Single-label vs. multi-label TC

- TC comes in two very different variants:
  - Single-label TC (SL): when exactly one category must be assigned to each document. This is the task of approximating the target function $\Phi : \mathcal{D} \rightarrow \mathcal{C}$ by means of a classifier $\hat{\Phi} : \mathcal{D} \rightarrow \mathcal{C}$
    - An important special case is when $m = 2$ (binary TC); typically, this means deciding whether c_j or $\overline{c}_j$ is the case.
  - Multi-label TC (ML): when any number $\{0, \ldots, m\}$ of categories can be assigned to each document. This is the task of approximating a target function $\Phi : \mathcal{D} \rightarrow \mathcal{P}(\mathcal{C})$ by means of a classifier $\hat{\Phi} : \mathcal{D} \rightarrow \mathcal{P}(\mathcal{C})$
- In either case, we will often indicate a target function with the alternative notation $\Phi : \mathcal{D} \times \mathcal{C} \rightarrow \{-1, +1\}$. Accordingly, a document d_i is called a positive example of c_j if $\Phi(d_i, c_j) = +1$, and a negative example of c_j if $\Phi(d_i, c_j) = -1$.

Single-label vs. multi-label TC

- TC comes in two very different variants:
  - Single-label TC (SL): when exactly one category must be assigned to each document. This is the task of approximating the target function $\Phi : \mathcal{D} \rightarrow \mathcal{C}$ by means of a classifier $\hat{\Phi} : \mathcal{D} \rightarrow \mathcal{C}$
    - An important special case is when $m = 2$ (binary TC); typically, this means deciding whether c_j or $\overline{c}_j$ is the case.
  - Multi-label TC (ML): when any number $\{0, \ldots, m\}$ of categories can be assigned to each document. This is the task of approximating a target function $\Phi : \mathcal{D} \rightarrow \mathcal{P}(\mathcal{C})$ by means of a classifier $\hat{\Phi} : \mathcal{D} \rightarrow \mathcal{P}(\mathcal{C})$
- In either case, we will often indicate a target function with the alternative notation $\Phi : \mathcal{D} \times \mathcal{C} \rightarrow \{-1, +1\}$. Accordingly, a document d_i is called a positive example of c_j if $\Phi(d_i, c_j) = +1$, and a negative example of c_j if $\Phi(d_i, c_j) = -1$.

Single-label vs. multi-label TC

- TC comes in two very different variants:
  - Single-label TC (SL): when exactly one category must be assigned to each document. This is the task of approximating the target function $\Phi : \mathcal{D} \rightarrow \mathcal{C}$ by means of a classifier $\hat{\Phi} : \mathcal{D} \rightarrow \mathcal{C}$
    - An important special case is when $m = 2$ (binary TC); typically, this means deciding whether c_j or $\overline{c}_j$ is the case.
  - Multi-label TC (ML): when any number $\{0, \ldots, m\}$ of categories can be assigned to each document. This is the task of approximating a target function $\Phi : \mathcal{D} \rightarrow \mathcal{P}(\mathcal{C})$ by means of a classifier $\hat{\Phi} : \mathcal{D} \rightarrow \mathcal{P}(\mathcal{C})$
- In either case, we will often indicate a target function with the alternative notation $\Phi : \mathcal{D} \times \mathcal{C} \rightarrow \{-1, +1\}$. Accordingly, a document d_i is called a positive example of c_j if $\Phi(d_i, c_j) = +1$, and a negative example of c_j if $\Phi(d_i, c_j) = -1$.

Single-label vs. multi-label TC

- TC comes in two very different variants:
  - Single-label TC (SL): when exactly one category must be assigned to each document. This is the task of approximating the target function $\Phi : \mathcal{D} \rightarrow \mathcal{C}$ by means of a classifier $\hat{\Phi} : \mathcal{D} \rightarrow \mathcal{C}$
    - An important special case is when $m = 2$ (binary TC); typically, this means deciding whether c_j or $\overline{c}_j$ is the case.
  - Multi-label TC (ML): when any number $\{0, \ldots, m\}$ of categories can be assigned to each document. This is the task of approximating a target function $\Phi : \mathcal{D} \rightarrow \mathcal{P}(\mathcal{C})$ by means of a classifier $\hat{\Phi} : \mathcal{D} \rightarrow \mathcal{P}(\mathcal{C})$
- In either case, we will often indicate a target function with the alternative notation $\Phi : \mathcal{D} \times \mathcal{C} \rightarrow \{-1, +1\}$. Accordingly, a document d_i is called a positive example of c_j if $\Phi(d_i, c_j) = +1$, and a negative example of c_j if $\Phi(d_i, c_j) = -1$.

Single-label vs. multi-label TC (cont'd)

- SL is more general than ML, since one needs only transform a ML problem under $\{c_1, \ldots, c_m\}$ into m independent binary problems under $\{c_j, \overline{c}_j\}$, for $j = 1, \ldots, m$.
- We will thus take a classifier for c_j to be a function $\hat{\Phi}_j : \mathcal{D} \rightarrow \{-1, +1\}$ that approximates an unknown function $\Phi_j : \mathcal{D} \rightarrow \{-1, +1\}$ (where $+1$ means membership in c_j and -1 means membership in $\overline{c}_j$).
- We will focus on the binary case, since
  - it is the most common in TC applications;
  - SL problems can anyway be solved by techniques similar to the ones we will discuss here, unless otherwise noted.

Single-label vs. multi-label TC (cont'd)

- SL is more general than ML, since one needs only transform a ML problem under $\{c_1, \ldots, c_m\}$ into m independent binary problems under $\{c_j, \overline{c}_j\}$, for $j = 1, \ldots, m$.
- We will thus take a classifier for c_j to be a function $\hat{\Phi}_j : \mathcal{D} \rightarrow \{-1, +1\}$ that approximates an unknown function $\Phi_j : \mathcal{D} \rightarrow \{-1, +1\}$ (where $+1$ means membership in c_j and -1 means membership in $\overline{c}_j$).
- We will focus on the binary case, since
  - it is the most common in TC applications;
  - SL problems can anyway be solved by techniques similar to the ones we will discuss here, unless otherwise noted.

Single-label vs. multi-label TC (cont'd)

- SL is more general than ML, since one needs only transform a ML problem under $\{c_1, \ldots, c_m\}$ into m independent binary problems under $\{c_j, \overline{c}_j\}$, for $j = 1, \ldots, m$.
- We will thus take a classifier for c_j to be a function $\hat{\Phi}_j : \mathcal{D} \rightarrow \{-1, +1\}$ that approximates an unknown function $\Phi_j : \mathcal{D} \rightarrow \{-1, +1\}$ (where $+1$ means membership in c_j and -1 means membership in $\overline{c}_j$).
- We will focus on the binary case, since
  - it is the most common in TC applications;
  - SL problems can anyway be solved by techniques similar to the ones we will discuss here, unless otherwise noted.

Category- and document-pivoted classification

We may want to apply a ML classifier in two alternative ways:

- Given $d_i \in \mathcal{D}$, find all the $c_j \in \mathcal{C}$ under which it should be filed (document-pivoted classification – DPC)
- Given $c_j \in \mathcal{C}$, find all the $d_i \in \mathcal{D}$ that should be filed under it (category-pivoted classification – CPC).

This distinction is only pragmatic but important, as the sets $\mathcal{D}$ and $\mathcal{C}$ are not always available in their entirety right from the start:

- DPC is suitable e.g. when documents become available one at a time (e.g. in e-mail filtering);
- CPC is suitable when new categories c_{m+1}, ... may be added to $\mathcal{C} = \{c_1, \ldots, c_m\}$ after a number of documents have already been classified under $\mathcal{C}$ (e.g. in patent classification).

We will focus on DPC since

- it the most common in TC applications;
- CPC can anyway be solved by techniques similar to the ones we will discuss here, unless otherwise noted.

Category- and document-pivoted classification

We may want to apply a ML classifier in two alternative ways:

- Given $d_i \in \mathcal{D}$, find all the $c_j \in \mathcal{C}$ under which it should be filed (document-pivoted classification – DPC)
- Given $c_j \in \mathcal{C}$, find all the $d_i \in \mathcal{D}$ that should be filed under it (category-pivoted classification – CPC).

This distinction is only pragmatic but important, as the sets $\mathcal{D}$ and $\mathcal{C}$ are not always available in their entirety right from the start:

- DPC is suitable e.g. when documents become available one at a time (e.g. in e-mail filtering);
- CPC is suitable when new categories c_{m+1}, ... may be added to $\mathcal{C} = \{c_1, \ldots, c_m\}$ after a number of documents have already been classified under $\mathcal{C}$ (e.g. in patent classification).

We will focus on DPC since

- it the most common in TC applications;
- CPC can anyway be solved by techniques similar to the ones we will discuss here, unless otherwise noted.

Category- and document-pivoted classification

We may want to apply a ML classifier in two alternative ways:

- Given $d_i \in \mathcal{D}$, find all the $c_j \in \mathcal{C}$ under which it should be filed (document-pivoted classification – DPC)
- Given $c_j \in \mathcal{C}$, find all the $d_i \in \mathcal{D}$ that should be filed under it (category-pivoted classification – CPC).

This distinction is only pragmatic but important, as the sets $\mathcal{D}$ and $\mathcal{C}$ are not always available in their entirety right from the start:

- DPC is suitable e.g. when documents become available one at a time (e.g. in e-mail filtering);
- CPC is suitable when new categories c_{m+1}, ... may be added to $\mathcal{C} = \{c_1, \ldots, c_m\}$ after a number of documents have already been classified under $\mathcal{C}$ (e.g. in patent classification).

We will focus on DPC since

- it the most common in TC applications;
- CPC can anyway be solved by techniques similar to the ones we will discuss here, unless otherwise noted.

Category- and document-pivoted classification

We may want to apply a ML classifier in two alternative ways:

- Given $d_i \in \mathcal{D}$, find all the $c_j \in \mathcal{C}$ under which it should be filed (document-pivoted classification – DPC)
- Given $c_j \in \mathcal{C}$, find all the $d_i \in \mathcal{D}$ that should be filed under it (category-pivoted classification – CPC).

This distinction is only pragmatic but important, as the sets $\mathcal{D}$ and $\mathcal{C}$ are not always available in their entirety right from the start:

- DPC is suitable e.g. when documents become available one at a time (e.g. in e-mail filtering);
- CPC is suitable when new categories c_{m+1}, ... may be added to $\mathcal{C} = \{c_1, \ldots, c_m\}$ after a number of documents have already been classified under $\mathcal{C}$ (e.g. in patent classification).

We will focus on DPC since

- it the most common in TC applications;
- CPC can anyway be solved by techniques similar to the ones we will discuss here, unless otherwise noted.

Category- and document-pivoted classification

We may want to apply a ML classifier in two alternative ways:

- Given $d_i \in \mathcal{D}$, find all the $c_j \in \mathcal{C}$ under which it should be filed (document-pivoted classification – DPC)
- Given $c_j \in \mathcal{C}$, find all the $d_i \in \mathcal{D}$ that should be filed under it (category-pivoted classification – CPC).

This distinction is only pragmatic but important, as the sets $\mathcal{D}$ and $\mathcal{C}$ are not always available in their entirety right from the start:

- DPC is suitable e.g. when documents become available one at a time (e.g. in e-mail filtering);
- CPC is suitable when new categories c_{m+1}, ... may be added to $\mathcal{C} = \{c_1, \ldots, c_m\}$ after a number of documents have already been classified under $\mathcal{C}$ (e.g. in patent classification).

We will focus on DPC since

- it the most common in TC applications;
- CPC can anyway be solved by techniques similar to the ones we will discuss here, unless otherwise noted.

"Hard" vs. "soft" classification

- Fully automated classifiers need to take a "hard", binary decision for each pair $\langle d_i, c_j \rangle$. Semi-automated, "interactive" classifiers are instead obtained by allowing for "soft" (i.e. real-valued) decisions:
  1. Given $d_i \in \mathcal{D}$ a system might rank the categories in $\mathcal{C} = \{c_1, \ldots, c_m\}$ according to their estimated appropriateness to d_i (category-ranking TC) [3]
  2. Given $c_j \in \mathcal{C}$ a system might rank the documents in $\mathcal{D}$ according to their estimated appropriateness to c_j (document-ranking TC).
- Such ranked lists would be of great help to a human expert in charge of taking the final classification decision, since she could thus only examine the items at the top of the list.
- Semi-automated classifiers are useful especially in critical applications in which the effectiveness of a fully automated system may be significantly lower than that of a human expert. We will mostly deal with "hard" classification.

"Hard" vs. "soft" classification

- Fully automated classifiers need to take a "hard", binary decision for each pair $\langle d_i, c_j \rangle$. Semi-automated, "interactive" classifiers are instead obtained by allowing for "soft" (i.e. real-valued) decisions:
  1. Given $d_i \in \mathcal{D}$ a system might rank the categories in $\mathcal{C} = \{c_1, \ldots, c_m\}$ according to their estimated appropriateness to d_i (category-ranking TC) [3]
  2. Given $c_j \in \mathcal{C}$ a system might rank the documents in $\mathcal{D}$ according to their estimated appropriateness to c_j (document-ranking TC).
- Such ranked lists would be of great help to a human expert in charge of taking the final classification decision, since she could thus only examine the items at the top of the list.
- Semi-automated classifiers are useful especially in critical applications in which the effectiveness of a fully automated system may be significantly lower than that of a human expert. We will mostly deal with "hard" classification.

"Hard" vs. "soft" classification

- Fully automated classifiers need to take a "hard", binary decision for each pair $\langle d_i, c_j \rangle$. Semi-automated, "interactive" classifiers are instead obtained by allowing for "soft" (i.e. real-valued) decisions:
  1. Given $d_i \in \mathcal{D}$ a system might rank the categories in $\mathcal{C} = \{c_1, \ldots, c_m\}$ according to their estimated appropriateness to d_i (category-ranking TC) [3]
  2. Given $c_j \in \mathcal{C}$ a system might rank the documents in $\mathcal{D}$ according to their estimated appropriateness to c_j (document-ranking TC).
- Such ranked lists would be of great help to a human expert in charge of taking the final classification decision, since she could thus only examine the items at the top of the list.
- Semi-automated classifiers are useful especially in critical applications in which the effectiveness of a fully automated system may be significantly lower than that of a human expert. We will mostly deal with "hard" classification.

"Hard" vs. "soft" classification

- Fully automated classifiers need to take a "hard", binary decision for each pair $\langle d_i, c_j \rangle$. Semi-automated, "interactive" classifiers are instead obtained by allowing for "soft" (i.e. real-valued) decisions:
  1. Given $d_i \in \mathcal{D}$ a system might rank the categories in $\mathcal{C} = \{c_1, \ldots, c_m\}$ according to their estimated appropriateness to d_i (category-ranking TC) [3]
  2. Given $c_j \in \mathcal{C}$ a system might rank the documents in $\mathcal{D}$ according to their estimated appropriateness to c_j (document-ranking TC).
- Such ranked lists would be of great help to a human expert in charge of taking the final classification decision, since she could thus only examine the items at the top of the list.
- Semi-automated classifiers are useful especially in critical applications in which the effectiveness of a fully automated system may be significantly lower than that of a human expert. We will mostly deal with "hard" classification.

Outline

- Since Maron's 1961 seminal work [18], TC has been used in a number of different applications. We will discuss in some detail
  1. automatic indexing for Boolean information retrieval
  2. document organization
  3. document filtering (e.g. e-mail filtering, spam filtering)
- Note that the borders between these classes of applications are imprecise and somehow artificial, and some of these applications may be considered special cases of others.

- Since Maron's 1961 seminal work [18], TC has been used in a number of different applications. We will discuss in some detail
  1. automatic indexing for Boolean information retrieval
  2. document organization
  3. document filtering (e.g. e-mail filtering, spam filtering)
- Note that the borders between these classes of applications are imprecise and somehow artificial, and some of these applications may be considered special cases of others.

Automatic indexing for Boolean information retrieval

- The application that spawned most of the early research in TC is that of automatic document indexing for use in Boolean IR systems. In these systems, each document is assigned one or more keywords belonging to a controlled dictionary. Usually, this is performed by trained human annotators, and is thus a costly activity.
- If the entries in the controlled dictionary are viewed as categories, document indexing is an instance of TC.
- This is a multi-label task, and document-pivoted classification ($\rightarrow$ documents classified as they become available) is used
- This form of automated indexing may also be viewed as a form of automated metadata generation (or ontology learning), which is going to be very important for the "Semantic Web".

Automatic indexing for Boolean information retrieval

- The application that spawned most of the early research in TC is that of automatic document indexing for use in Boolean IR systems. In these systems, each document is assigned one or more keywords belonging to a controlled dictionary. Usually, this is performed by trained human annotators, and is thus a costly activity.
- If the entries in the controlled dictionary are viewed as categories, document indexing is an instance of TC.
- This is a multi-label task, and document-pivoted classification ($\rightarrow$ documents classified as they become available) is used
- This form of automated indexing may also be viewed as a form of automated metadata generation (or ontology learning), which is going to be very important for the "Semantic Web".

Automatic indexing for Boolean information retrieval

- The application that spawned most of the early research in TC is that of automatic document indexing for use in Boolean IR systems. In these systems, each document is assigned one or more keywords belonging to a controlled dictionary. Usually, this is performed by trained human annotators, and is thus a costly activity.
- If the entries in the controlled dictionary are viewed as categories, document indexing is an instance of TC.
- This is a multi-label task, and document-pivoted classification ($\rightarrow$ documents classified as they become available) is used
- This form of automated indexing may also be viewed as a form of automated metadata generation (or ontology learning), which is going to be very important for the "Semantic Web".

Document organization

- Many issues pertaining to document organization and filing, be it for purposes of personal organization or document repository structuring, may be addressed by automatic classification techniques. Possible instances are:
  - classifying "classified" ads;
  - classifying "incoming" articles at a newspaper;
  - classifying patents for easing their later retrieval;
  - grouping conference papers into sessions;
  - assigning a paper to review to the right expert reviewer.

Document filtering

- Document filtering (DF) is the classification of a dynamic stream of incoming documents dispatched in an asynchronous way by an information producer to an information consumer.
- A typical example is a newsfeed (the information producer is a news agency and the information consumer is a newspaper). In this case, the DF system should discard the documents the consumer is not likely to be interested in. Other examples are spam filters.
- A DF system may be installed
  - at the producer end, in which case its role is to route the information to the interested consumers only (selective dissemination of information)
  - at the consumer end, in which case its role is to block the delivery of information deemed uninteresting to the consumer. This is the most frequent case (e.g. in filtering spam or other unsuitable content).

Document filtering

- Document filtering (DF) is the classification of a dynamic stream of incoming documents dispatched in an asynchronous way by an information producer to an information consumer.
- A typical example is a newsfeed (the information producer is a news agency and the information consumer is a newspaper). In this case, the DF system should discard the documents the consumer is not likely to be interested in. Other examples are spam filters.
- A DF system may be installed
  - at the producer end, in which case its role is to route the information to the interested consumers only (selective dissemination of information)
  - at the consumer end, in which case its role is to block the delivery of information deemed uninteresting to the consumer. This is the most frequent case (e.g. in filtering spam or other unsuitable content).

Document filtering

- Document filtering (DF) is the classification of a dynamic stream of incoming documents dispatched in an asynchronous way by an information producer to an information consumer.
- A typical example is a newsfeed (the information producer is a news agency and the information consumer is a newspaper). In this case, the DF system should discard the documents the consumer is not likely to be interested in. Other examples are spam filters.
- A DF system may be installed
  - at the producer end, in which case its role is to route the information to the interested consumers only (selective dissemination of information)
  - at the consumer end, in which case its role is to block the delivery of information deemed uninteresting to the consumer. This is the most frequent case (e.g. in filtering spam or other unsuitable content).

Document filtering

- Document filtering (DF) is the classification of a dynamic stream of incoming documents dispatched in an asynchronous way by an information producer to an information consumer.
- A typical example is a newsfeed (the information producer is a news agency and the information consumer is a newspaper). In this case, the DF system should discard the documents the consumer is not likely to be interested in. Other examples are spam filters.
- A DF system may be installed
  - at the producer end, in which case its role is to route the information to the interested consumers only (selective dissemination of information)
  - at the consumer end, in which case its role is to block the delivery of information deemed uninteresting to the consumer. This is the most frequent case (e.g. in filtering spam or other unsuitable content).

Other applications

Other (sometimes esoteric) applications are:

- Author (or author's gender) identification for documents of disputed paternity [11];
- Automatic identification of text genre [6, 15] or Web page genre [19];
- Polarity detection (aka "sentiment classification") [1, 21];
- Image annotation via caption analysis [22];
- Speech classification via speech recognition + TC [20, 23];
- Automated survey coding [8, 17];
- Language identification [9].

Other applications

Other (sometimes esoteric) applications are:

- Author (or author's gender) identification for documents of disputed paternity [11];
- Automatic identification of text genre [6, 15] or Web page genre [19];
- Polarity detection (aka "sentiment classification") [1, 21];
- Image annotation via caption analysis [22];
- Speech classification via speech recognition + TC [20, 23];
- Automated survey coding [8, 17];
- Language identification [9].

Other applications

Other (sometimes esoteric) applications are:

- Author (or author's gender) identification for documents of disputed paternity [11];
- Automatic identification of text genre [6, 15] or Web page genre [19];
- Polarity detection (aka "sentiment classification") [1, 21];
- Image annotation via caption analysis [22];
- Speech classification via speech recognition + TC [20, 23];
- Automated survey coding [8, 17];
- Language identification [9].

Other applications

Other (sometimes esoteric) applications are:

- Author (or author's gender) identification for documents of disputed paternity [11];
- Automatic identification of text genre [6, 15] or Web page genre [19];
- Polarity detection (aka "sentiment classification") [1, 21];
- Image annotation via caption analysis [22];
- Speech classification via speech recognition + TC [20, 23];
- Automated survey coding [8, 17];
- Language identification [9].

Other applications

Other (sometimes esoteric) applications are:

- Author (or author's gender) identification for documents of disputed paternity [11];
- Automatic identification of text genre [6, 15] or Web page genre [19];
- Polarity detection (aka "sentiment classification") [1, 21];
- Image annotation via caption analysis [22];
- Speech classification via speech recognition + TC [20, 23];
- Automated survey coding [8, 17];
- Language identification [9].

Outline

- In the '80s, the typical approach used for the construction of TC systems involved hand-crafting an expert system consisting of a set of rules, one per category, of the form

$$\textbf{if } \langle \textit{DNF formula} \rangle \textbf{ then } \langle \textit{category} \rangle \textbf{ else } \neg \langle \textit{category} \rangle$$

- A DNF formula is a disjunction of conjunctive clauses; the document is thus classified under $\langle \textit{category} \rangle$ iff it satisfies at least one of the clauses.
- The drawback of this "manual" approach is the knowledge acquisition bottleneck : since rules must be manually defined, building a classifier is expensive, and if the set of categories is updated or the classifier is ported to a different domain, other manual work has to be done.

- In the '80s, the typical approach used for the construction of TC systems involved hand-crafting an expert system consisting of a set of rules, one per category, of the form

$$\textbf{if } \langle \textit{DNF formula} \rangle \textbf{ then } \langle \textit{category} \rangle \textbf{ else } \neg \langle \textit{category} \rangle$$

- A DNF formula is a disjunction of conjunctive clauses; the document is thus classified under $\langle \textit{category} \rangle$ iff it satisfies at least one of the clauses.
- The drawback of this "manual" approach is the knowledge acquisition bottleneck : since rules must be manually defined, building a classifier is expensive, and if the set of categories is updated or the classifier is ported to a different domain, other manual work has to be done.

- In the '80s, the typical approach used for the construction of TC systems involved hand-crafting an expert system consisting of a set of rules, one per category, of the form

$$\textbf{if } \langle \textit{DNF formula} \rangle \textbf{ then } \langle \textit{category} \rangle \textbf{ else } \neg \langle \textit{category} \rangle$$

- A DNF formula is a disjunction of conjunctive clauses; the document is thus classified under $\langle \textit{category} \rangle$ iff it satisfies at least one of the clauses.
- The drawback of this "manual" approach is the knowledge acquisition bottleneck : since rules must be manually defined, building a classifier is expensive, and if the set of categories is updated or the classifier is ported to a different domain, other manual work has to be done.

if ((wheat & farm) **or**
(wheat & commodity) **or**
(bushels & export) **or**
(wheat & tonnes) **or**
(wheat & winter & ¬ soft)) **then** CEREALS **else** ¬ CEREALS

Figure: Example of a classification rule.

		expert judgments	
		CEREALS	¬ CEREALS
classifier judgments	CEREALS	73	8
	¬ CEREALS	14	3577

Figure: A contingency table for the classification rule above.

- Since the early '90s, the machine learning approach to the construction of TC systems has become dominant. A general inductive process automatically builds a classifier for a category c_j by "observing" the characteristics of a set of documents previously classified under c_j or $\overline{c}_j$ by a domain expert. This is an instance of supervised learning.
- Advantages of this approach :
  1. The engineering effort goes towards the construction not of a classifier, but of an automatic builder of classifiers (learner) $\longrightarrow$ if the set of categories is updated, or if the system is ported to a different domain, all that is needed is a different set of manually classified documents.
  2. Domain expertise (for labelling), and not knowledge engineering expertise, is needed; this is advantageous, since it is easier to characterize a concept "ostensively" than "intensionally".
  3. Sometimes the preclassified documents are already available.
  4. The effectiveness achievable nowadays by these classifiers exceeds that of hand-crafted classifiers, and sometimes rivals that of human annotators.

- Since the early '90s, the machine learning approach to the construction of TC systems has become dominant. A general inductive process automatically builds a classifier for a category c_j by "observing" the characteristics of a set of documents previously classified under c_j or $\overline{c}_j$ by a domain expert. This is an instance of supervised learning.
- Advantages of this approach :
  1. The engineering effort goes towards the construction not of a classifier, but of an automatic builder of classifiers (learner) $\longrightarrow$ if the set of categories is updated, or if the system is ported to a different domain, all that is needed is a different set of manually classified documents.
  2. Domain expertise (for labelling), and not knowledge engineering expertise, is needed; this is advantageous, since it is easier to characterize a concept "ostensively" than "intensionally".
  3. Sometimes the preclassified documents are already available.
  4. The effectiveness achievable nowadays by these classifiers exceeds that of hand-crafted classifiers, and sometimes rivals that of human annotators.

- Since the early '90s, the machine learning approach to the construction of TC systems has become dominant. A general inductive process automatically builds a classifier for a category c_j by "observing" the characteristics of a set of documents previously classified under c_j or $\overline{c}_j$ by a domain expert. This is an instance of supervised learning.
- Advantages of this approach :
  1. The engineering effort goes towards the construction not of a classifier, but of an automatic builder of classifiers (learner) $\longrightarrow$ if the set of categories is updated, or if the system is ported to a different domain, all that is needed is a different set of manually classified documents.
  2. Domain expertise (for labelling), and not knowledge engineering expertise, is needed; this is advantageous, since it is easier to characterize a concept "ostensively" than "intensionally".
  3. Sometimes the preclassified documents are already available.
  4. The effectiveness achievable nowadays by these classifiers exceeds that of hand-crafted classifiers, and sometimes rivals that of human annotators.

- Since the early '90s, the machine learning approach to the construction of TC systems has become dominant. A general inductive process automatically builds a classifier for a category c_j by "observing" the characteristics of a set of documents previously classified under c_j or $\overline{c}_j$ by a domain expert. This is an instance of supervised learning.
- Advantages of this approach :
  1. The engineering effort goes towards the construction not of a classifier, but of an automatic builder of classifiers (learner) $\longrightarrow$ if the set of categories is updated, or if the system is ported to a different domain, all that is needed is a different set of manually classified documents.
  2. Domain expertise (for labelling), and not knowledge engineering expertise, is needed; this is advantageous, since it is easier to characterize a concept "ostensively" than "intensionally".
  3. Sometimes the preclassified documents are already available.
  4. The effectiveness achievable nowadays by these classifiers exceeds that of hand-crafted classifiers, and sometimes rivals that of human annotators.

- Since the early '90s, the machine learning approach to the construction of TC systems has become dominant. A general inductive process automatically builds a classifier for a category c_j by "observing" the characteristics of a set of documents previously classified under c_j or $\overline{c}_j$ by a domain expert. This is an instance of supervised learning.
- Advantages of this approach :
  1. The engineering effort goes towards the construction not of a classifier, but of an automatic builder of classifiers (learner) $\longrightarrow$ if the set of categories is updated, or if the system is ported to a different domain, all that is needed is a different set of manually classified documents.
  2. Domain expertise (for labelling), and not knowledge engineering expertise, is needed; this is advantageous, since it is easier to characterize a concept "ostensively" than "intensionally".
  3. Sometimes the preclassified documents are already available.
  4. The effectiveness achievable nowadays by these classifiers exceeds that of hand-crafted classifiers, and sometimes rivals that of human annotators.

Training set and test set

- The ML approach relies on the application of a train-and-test approach to a labelled corpus $\Omega = \{d_1, \ldots, d_{|\Omega|}\} \subset \mathcal{D}$, i.e. a set of documents previously classified under $\mathcal{C} = \{c_1, \ldots, c_m\}$.
- This means that the values of the total function $\Phi : \mathcal{D} \times \mathcal{C} \to \{-1, +1\}$ are known for every pair $\langle d_i, c_j \rangle \in \Omega \times \mathcal{C}$. The labelled corpus thus constitutes a "glimpse" of the ground truth.
- In the train-and-test approach Ω is partitioned into two sets:
  - training set $Tr = \{d_1, \ldots, d_{|Tr|}\}$: the set from which the classifier is inductively built;
  - test set $Te = \{d_{|Tr|+1}, \ldots, d_{|\Omega|}\}$: the set used for testing the effectiveness of the classifier just built.
- The documents in Te cannot participate in any way in the inductive construction of the classifiers!

Training set and test set

- The ML approach relies on the application of a train-and-test approach to a labelled corpus $\Omega = \{d_1, \ldots, d_{|\Omega|}\} \subset \mathcal{D}$, i.e. a set of documents previously classified under $\mathcal{C} = \{c_1, \ldots, c_m\}$.
- This means that the values of the total function $\Phi : \mathcal{D} \times \mathcal{C} \to \{-1, +1\}$ are known for every pair $\langle d_i, c_j \rangle \in \Omega \times \mathcal{C}$. The labelled corpus thus constitutes a "glimpse" of the ground truth.
- In the train-and-test approach Ω is partitioned into two sets:
  - training set $Tr = \{d_1, \ldots, d_{|Tr|}\}$: the set from which the classifier is inductively built;
  - test set $Te = \{d_{|Tr|+1}, \ldots, d_{|\Omega|}\}$: the set used for testing the effectiveness of the classifier just built.
- The documents in Te cannot participate in any way in the inductive construction of the classifiers!

Training set and test set

- The ML approach relies on the application of a train-and-test approach to a labelled corpus $\Omega = \{d_1, \ldots, d_{|\Omega|}\} \subset \mathcal{D}$, i.e. a set of documents previously classified under $\mathcal{C} = \{c_1, \ldots, c_m\}$.
- This means that the values of the total function $\Phi : \mathcal{D} \times \mathcal{C} \to \{-1, +1\}$ are known for every pair $\langle d_i, c_j \rangle \in \Omega \times \mathcal{C}$. The labelled corpus thus constitutes a "glimpse" of the ground truth.
- In the train-and-test approach Ω is partitioned into two sets:
  - training set $Tr = \{d_1, \ldots, d_{|Tr|}\}$: the set from which the classifier is inductively built;
  - test set $Te = \{d_{|Tr|+1}, \ldots, d_{|\Omega|}\}$: the set used for testing the effectiveness of the classifier just built.
- The documents in Te cannot participate in any way in the inductive construction of the classifiers!

- An alternative to this train-and-test approach is the *k*-fold cross-validation approach; this is especially used when the training corpus is small:
  1. Ω is partitioned into k disjoint sets $\Omega_1, \ldots, \Omega_k$
  2. for each $i \in \{1, \ldots, k\}$, a train-and-test run is conducted by using Ω/Ω_i as the training set and Ω_i as the test set;
  3. effectiveness is computed on the union of the k test sets Ω_i.
- The case $k = |\Omega|$ is called leave-one-out cross-validation.

- If the learned classifier is parametric, its internal parameters should be optimized by testing which values of the parameters yield the best effectiveness.
- In this case, the train-and-test protocol is modified as follows:
  1. a small subset of Tr, called the validation set (denoted by Va), is identified;
  2. the classifier is trained on Tr/Va;
  3. repeated tests of the classifier, with different values for the parameters, are performed on Va so as to find the optimal parameter values
  4. the final classifier is retrained on the entire Tr, using the optimal parameter values found in the previous step.

The text classification process

- The text classification process consists of the following phases:
  1. Document Indexing. This takes as input the training, validation, and test documents, and outputs internal representations for them. This is accomplished by techniques from the traditions of IR and information theory.
  2. Classifier Learning. This takes as input the representations of the training and validation documents and outputs a classifier. This is accomplished by ML techniques.
  3. Classifier Evaluation. This takes as input the results of the classification of the test set, and is mostly accomplished by evaluation techniques belonging to both the IR and the ML tradition. documents.
- The documents that will be classified during the operational phase can be likened to test documents.

The text classification process

- The text classification process consists of the following phases:
  1. Document Indexing. This takes as input the training, validation, and test documents, and outputs internal representations for them. This is accomplished by techniques from the traditions of IR and information theory.
  2. Classifier Learning. This takes as input the representations of the training and validation documents and outputs a classifier. This is accomplished by ML techniques.
  3. Classifier Evaluation. This takes as input the results of the classification of the test set, and is mostly accomplished by evaluation techniques belonging to both the IR and the ML tradition. documents.
- The documents that will be classified during the operational phase can be likened to test documents.

The text classification process

- The text classification process consists of the following phases:
  1. Document Indexing. This takes as input the training, validation, and test documents, and outputs internal representations for them. This is accomplished by techniques from the traditions of IR and information theory.
  2. Classifier Learning. This takes as input the representations of the training and validation documents and outputs a classifier. This is accomplished by ML techniques.
  3. Classifier Evaluation. This takes as input the results of the classification of the test set, and is mostly accomplished by evaluation techniques belonging to both the IR and the ML tradition. documents.
- The documents that will be classified during the operational phase can be likened to test documents.

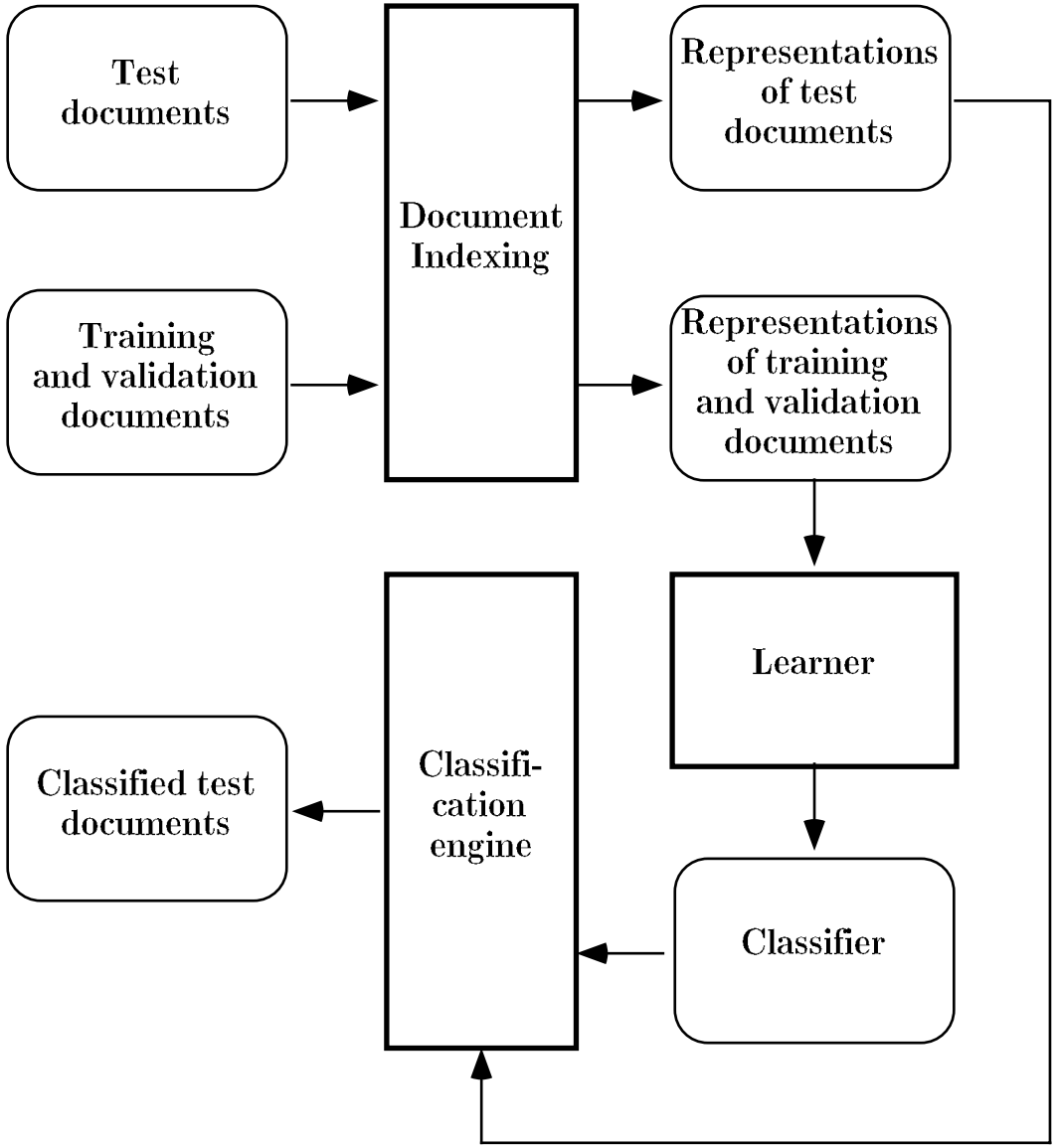

Figure: Architecture of a text classification system.

Outline

- As in most of IR, a document d_i is represented as a (sparse) vector of weights $\mathbf{d}_i = \langle w_{1i}, \ldots, w_{ni} \rangle$, where $n = |T|$ is the number of terms (usually: words) that occur at least α times in Tr. Weights w_{ki} may belong
  - to $\{0,1\}$ (the set-of-words approach), representing the presence or absence of t_k in d_i.
  - to $[0,1]$ (the bag-of-words approach), representing the (quantitative) contribution of t_k to the semantics of d_i.

  The choice between the two is dictated by the learning algorithm employed.
- Before weighting, the text is usually pre-processed by
  - removing stop words (i.e. topic-neutral words).
  - stemming, i.e. conflating different inflectional forms of the same word into a single class. This positively affects the dimensionality of the term space and the level of independence among terms.

- As in most of IR, a document d_i is represented as a (sparse) vector of weights $\mathbf{d}_i = \langle w_{1i}, \ldots, w_{ni} \rangle$, where $n = |T|$ is the number of terms (usually: words) that occur at least α times in Tr. Weights w_{ki} may belong
  - to $\{0,1\}$ (the set-of-words approach), representing the presence or absence of t_k in d_i.
  - to $[0,1]$ (the bag-of-words approach), representing the (quantitative) contribution of t_k to the semantics of d_i.

  The choice between the two is dictated by the learning algorithm employed.
- Before weighting, the text is usually pre-processed by
  - removing stop words (i.e. topic-neutral words).
  - stemming, i.e. conflating different inflectional forms of the same word into a single class. This positively affects the dimensionality of the term space and the level of independence among terms.

- As in most of IR, a document d_i is represented as a (sparse) vector of weights $\mathbf{d}_i = \langle w_{1i}, \ldots, w_{ni} \rangle$, where $n = |T|$ is the number of terms (usually: words) that occur at least α times in Tr. Weights w_{ki} may belong
  - to $\{0,1\}$ (the set-of-words approach), representing the presence or absence of t_k in d_i.
  - to $[0,1]$ (the bag-of-words approach), representing the (quantitative) contribution of t_k to the semantics of d_i.

  The choice between the two is dictated by the learning algorithm employed.
- Before weighting, the text is usually pre-processed by
  - removing stop words (i.e. topic-neutral words).
  - stemming, i.e. conflating different inflectional forms of the same word into a single class. This positively affects the dimensionality of the term space and the level of independence among terms.

- As in most of IR, a document d_i is represented as a (sparse) vector of weights $\mathbf{d}_i = \langle w_{1i}, \ldots, w_{ni} \rangle$, where $n = |T|$ is the number of terms (usually: words) that occur at least α times in Tr. Weights w_{ki} may belong
  - to $\{0,1\}$ (the set-of-words approach), representing the presence or absence of t_k in d_i.
  - to $[0,1]$ (the bag-of-words approach), representing the (quantitative) contribution of t_k to the semantics of d_i.

  The choice between the two is dictated by the learning algorithm employed.
- Before weighting, the text is usually pre-processed by
  - removing stop words (i.e. topic-neutral words).
  - stemming, i.e. conflating different inflectional forms of the same word into a single class. This positively affects the dimensionality of the term space and the level of independence among terms.

- Slightly more sophisticated notions of "terms", consisting of units larger than words (ULTWs), have sometimes been used:
  - (linguistically motivated) phrases [16], i.e. noun phrases or verb phrases;
  - head-modifier phrases [13], i.e. pairs consisting of a head (e.g. a noun) and its modifier (e.g. an adjective);
  - statistically motivated phrases [2], i.e. pairs (or triples) of words occurring contiguously in a statistically significant way.
- It has generally been found that ULTWs do not yield superior effectiveness wrt single words. The likely reason is that, although ULTWs have superior semantic qualities, they have inferior statistical qualities wrt single words.
- Wrt a word-only indexing language, a ULTWs-only indexing language has

> *"more terms, more synonymous or nearly synonymous terms, lower consistency of assignment (since synonymous terms are not assigned to the same documents), and lower document frequency for terms". [16]*

- In the [0,1] case, for determining the weight w_{ki} of term t_k in document d_i any IR weighting technique may be used, e.g.,

$$tfidf(t_k, d_i) = tf(t_k, d_i) \cdot idf(t_k)$$

where

- $tf(t_k, d_i) = \begin{cases} 1 + \log \#(t_k, d_i) & \text{if } \#(t_k, d_i) > 0 \\ 0 & \text{otherwise} \end{cases}$

  with $\#(t_k, d_i)$ the number of times t_k occurs in d_i (term frequency)
- $idf(t_k) = \log \frac{|Tr|}{\#_{Tr}(t_k)}$ (inverse document frequency).

- Cosine normalization is often applied to the weights resulting from *tfidf*, so as to account for document length: i.e.

$$w_{ki} = \frac{tfidf(t_k, d_i)}{\sqrt{\sum_{t_s \in T} (tfidf(t_s, d_i))^2}}$$

- A more modern, better performing alternative is BM25.

- In the [0,1] case, for determining the weight w_{ki} of term t_k in document d_i any IR weighting technique may be used, e.g.,

$$tfidf(t_k, d_i) = tf(t_k, d_i) \cdot idf(t_k)$$

where

- $tf(t_k, d_i) = \begin{cases} 1 + \log \#(t_k, d_i) & \text{if } \#(t_k, d_i) > 0 \\ 0 & \text{otherwise} \end{cases}$

  with $\#(t_k, d_i)$ the number of times t_k occurs in d_i (term frequency)
- $idf(t_k) = \log \frac{|Tr|}{\#_{Tr}(t_k)}$ (inverse document frequency).

- Cosine normalization is often applied to the weights resulting from *tfidf*, so as to account for document length: i.e.

$$w_{ki} = \frac{tfidf(t_k, d_i)}{\sqrt{\sum_{t_s \in T} (tfidf(t_s, d_i))^2}}$$

- A more modern, better performing alternative is BM25.

- In the [0,1] case, for determining the weight w_{ki} of term t_k in document d_i any IR weighting technique may be used, e.g.,

$$tfidf(t_k, d_i) = tf(t_k, d_i) \cdot idf(t_k)$$

where

- $$tf(t_k, d_i) = \begin{cases} 1 + \log \#(t_k, d_i) & \text{if } \#(t_k, d_i) > 0 \\ 0 & \text{otherwise} \end{cases}$$

  with $\#(t_k, d_i)$ the number of times t_k occurs in d_i (term frequency)
- $idf(t_k) = \log \frac{|Tr|}{\#_{Tr}(t_k)}$ (inverse document frequency).

- Cosine normalization is often applied to the weights resulting from *tfidf*, so as to account for document length: i.e.

$$w_{ki} = \frac{tfidf(t_k, d_i)}{\sqrt{\sum_{t_s \in T} (tfidf(t_s, d_i))^2}}$$

- A more modern, better performing alternative is BM25.

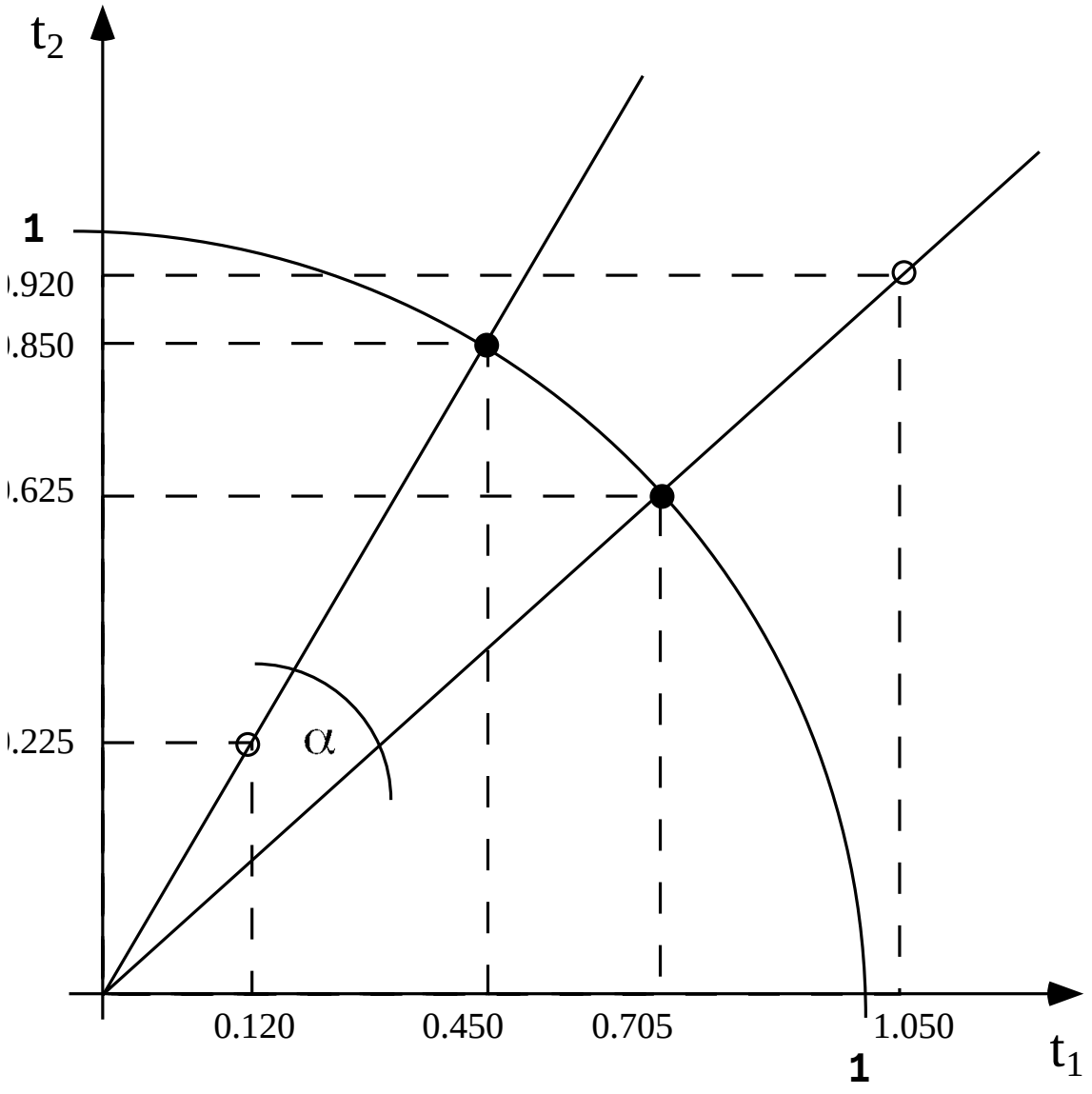

Figure: Document vectors in a 2-dimensional vector space.

- The choices discussed above (in particular: choosing words as terms, performing stop word removal and stemming) are adequate in TC applications related to topic identification, since content words tend to be indicative of topic, while stop words and morphological suffixes tend to be topic-neutral.
- When topic identification is not the issue, the choices may be different; for instance
  - in automated authorship attribution
  - in genre identification

  the decision to be taken is orthogonal to the topic of the document. In these applications people tend to use stop words plus other additional style-related features [12], such as e.g.
  - vocabulary richness (i.e. ratio between number of distinct words and total number of words)
  - average word length
  - average sentence length
  - ...

- The choices discussed above (in particular: choosing words as terms, performing stop word removal and stemming) are adequate in TC applications related to topic identification, since content words tend to be indicative of topic, while stop words and morphological suffixes tend to be topic-neutral.
- When topic identification is not the issue, the choices may be different; for instance
  - in automated authorship attribution
  - in genre identification

  the decision to be taken is orthogonal to the topic of the document. In these applications people tend to use stop words plus other additional style-related features [12], such as e.g.
  - vocabulary richness (i.e. ratio between number of distinct words and total number of words)
  - average word length
  - average sentence length
  - ...

Dimensionality reduction

Techniques for dimensionality reduction (DR) are often employed before weighting, in order to reduce the dimensionality of the vector space from $n = |T|$ to $n' \ll n$, since

- in TC the high dimensionality n of the term space (n may well be $O(10^5)$) may be problematic, as many sophisticated learning algorithms used for TC do not scale well to high values of n;
- DR reduces overfitting, i.e. the phenomenon by which a classifier is tuned also to the contingent characteristics of the training data, rather than just the constitutive characteristics of the category. For avoiding overfitting, the higher the dimensionality of the vector space is, the more training examples are needed, since
  - training examples act as constraints on the problem
  - features act as variables of the problem

  Overfitting is thus a symptom of an underconstrained problem.

Dimensionality reduction

Techniques for dimensionality reduction (DR) are often employed before weighting, in order to reduce the dimensionality of the vector space from $n = |T|$ to $n' \ll n$, since

- in TC the high dimensionality n of the term space (n may well be $O(10^5)$) may be problematic, as many sophisticated learning algorithms used for TC do not scale well to high values of n;
- DR reduces overfitting, i.e. the phenomenon by which a classifier is tuned also to the contingent characteristics of the training data, rather than just the constitutive characteristics of the category. For avoiding overfitting, the higher the dimensionality of the vector space is, the more training examples are needed, since
  - training examples act as constraints on the problem
  - features act as variables of the problem

  Overfitting is thus a symptom of an underconstrained problem.

Dimensionality reduction

Techniques for dimensionality reduction (DR) are often employed before weighting, in order to reduce the dimensionality of the vector space from $n = |T|$ to $n' \ll n$, since

- in TC the high dimensionality n of the term space (n may well be $O(10^5)$) may be problematic, as many sophisticated learning algorithms used for TC do not scale well to high values of n;
- DR reduces overfitting, i.e. the phenomenon by which a classifier is tuned also to the contingent characteristics of the training data, rather than just the constitutive characteristics of the category. For avoiding overfitting, the higher the dimensionality of the vector space is, the more training examples are needed, since
  - training examples act as constraints on the problem
  - features act as variables of the problem

  Overfitting is thus a symptom of an underconstrained problem.

Term selection

- Given a fixed $n' \ll n$, techniques for term selection (aka term space reduction – TSR) attempt to select, from the original set of n terms, the n' terms that, when used for document indexing, give the best effectiveness. TSR must try to simultaneously
  - avoid discarding possibly useful information;
  - avoid overfitting and enhance training-time and classification-time efficiency.
- Comparative experiments [29] have been carried out in order to determine the impact on classifier effectiveness of
  - the aggressivity $\frac{n}{n'}$ of the reduction
  - the TSR technique used

  as a function of the type of classifier used.

Term selection

- Given a fixed $n' \ll n$, techniques for term selection (aka term space reduction – TSR) attempt to select, from the original set of n terms, the n' terms that, when used for document indexing, give the best effectiveness. TSR must try to simultaneously
  - avoid discarding possibly useful information;
  - avoid overfitting and enhance training-time and classification-time efficiency.
- Comparative experiments [29] have been carried out in order to determine the impact on classifier effectiveness of
  - the aggressivity $\frac{n}{n'}$ of the reduction
  - the TSR technique used

  as a function of the type of classifier used.

- TSR is often tackled by "filtering" methods, i.e. methods based on "greedily" selecting the $n' \ll n$ top-scoring terms according to a predetermined numerical function that measures the "potential" of the term for the TC task.
- "Supervised" feature selection functions are usually employed for this, i.e. functions that learn a score for term t_k from the distribution of t_k across the categories in the training data.
- Most of these functions try to capture the intuition that the most valuable terms for classification under c_j are those that are distributed most differently in the sets of positive and negative examples of the category.

- TSR is often tackled by "filtering" methods, i.e. methods based on "greedily" selecting the $n' \ll n$ top-scoring terms according to a predetermined numerical function that measures the "potential" of the term for the TC task.
- "Supervised" feature selection functions are usually employed for this, i.e. functions that learn a score for term t_k from the distribution of t_k across the categories in the training data.
- Most of these functions try to capture the intuition that the most valuable terms for classification under c_j are those that are distributed most differently in the sets of positive and negative examples of the category.

- TSR is often tackled by "filtering" methods, i.e. methods based on "greedily" selecting the $n' \ll n$ top-scoring terms according to a predetermined numerical function that measures the "potential" of the term for the TC task.
- "Supervised" feature selection functions are usually employed for this, i.e. functions that learn a score for term t_k from the distribution of t_k across the categories in the training data.
- Most of these functions try to capture the intuition that the most valuable terms for classification under c_j are those that are distributed most differently in the sets of positive and negative examples of the category.

Function	Denoted by	Mathematical form
Information gain	$IG(t_k, c_j)$	$\sum_{x \in \{c_j, \overline{c}_j\}} \sum_{y \in \{t_k, \overline{t}_k\}} \hat{\Pr}(x, y) \cdot \log \frac{\hat{\Pr}(x, y)}{\hat{\Pr}(x) \cdot \hat{\Pr}(y)}$
Gain ratio	$GR(t_k, c_j)$	$\frac{IG(t_k, c_j)}{-\sum_{x \in \{c_j, \overline{c}_j\}} \hat{\Pr}(x) \log \hat{\Pr}(x)}$
Pointwise MI	$PMI(t_k, c_j)$	$\log \frac{\hat{\Pr}(t_k, c_j)}{\hat{\Pr}(t_k) \cdot \hat{\Pr}(c_j)}$
Chi-square	$\chi^2(t_k, c_j)$	$\lvert Tr \rvert \cdot \frac{[\hat{\Pr}(t_k, c_j) \cdot \hat{\Pr}(\overline{t}_k, \overline{c}_j) - \hat{\Pr}(t_k, \overline{c}_j) \cdot \hat{\Pr}(\overline{t}_k, c_j)]^2}{\hat{\Pr}(t_k) \cdot \hat{\Pr}(\overline{t}_k) \cdot \hat{\Pr}(c_j) \cdot \hat{\Pr}(\overline{c}_j)}$
Odds Ratio	$OR(t_k, c_j)$	$\log \frac{\hat{\Pr}(t_k \lvert c_j) \cdot \hat{\Pr}(\overline{t}_k \lvert \overline{c}_j)}{\hat{\Pr}(\overline{t}_k \lvert c_j) \cdot \hat{\Pr}(t_k \lvert \overline{c}_j)}$

- Note that all these functions have a value of 0 when t_k and c_j, viewed as random variables, are independent.
- Note also that all these functions (except for PMI) reward not only the positive, but also the negative correlation of t_k to c_j.

Function	Denoted by	Mathematical form
Information gain	$IG(t_k, c_j)$	$\sum_{x \in \{c_j, \bar{c}_j\}} \sum_{y \in \{t_k, \bar{t}_k\}} \hat{\Pr}(x, y) \cdot \log \frac{\hat{\Pr}(x, y)}{\hat{\Pr}(x) \cdot \hat{\Pr}(y)}$
Gain ratio	$GR(t_k, c_j)$	$\frac{IG(t_k, c_j)}{-\sum_{x \in \{c_j, \bar{c}_j\}} \hat{\Pr}(x) \log \hat{\Pr}(x)}$
Pointwise MI	$PMI(t_k, c_j)$	$\log \frac{\hat{\Pr}(t_k, c_j)}{\hat{\Pr}(t_k) \cdot \hat{\Pr}(c_j)}$
Chi-square	$\chi^2(t_k, c_j)$	$\lvert Tr \rvert \cdot \frac{[\hat{\Pr}(t_k, c_j) \cdot \hat{\Pr}(\bar{t}_k, \bar{c}_j) - \hat{\Pr}(t_k, \bar{c}_j) \cdot \hat{\Pr}(\bar{t}_k, c_j)]^2}{\hat{\Pr}(t_k) \cdot \hat{\Pr}(\bar{t}_k) \cdot \hat{\Pr}(c_j) \cdot \hat{\Pr}(\bar{c}_j)}$
Odds Ratio	$OR(t_k, c_j)$	$\log \frac{\hat{\Pr}(t_k \lvert c_j) \cdot \hat{\Pr}(\bar{t}_k \lvert \bar{c}_j)}{\hat{\Pr}(\bar{t}_k \lvert c_j) \cdot \hat{\Pr}(t_k \lvert \bar{c}_j)}$

- Note that all these functions have a value of 0 when t_k and c_j, viewed as random variables, are independent.
- Note also that all these functions (except for PMI) reward not only the positive, but also the negative correlation of t_k to c_j.

Function	Denoted by	Mathematical form
Information gain	$IG(t_k, c_j)$	$\sum_{x \in \{c_j, \overline{c}_j\}} \sum_{y \in \{t_k, \overline{t}_k\}} \hat{\Pr}(x, y) \cdot \log \frac{\hat{\Pr}(x, y)}{\hat{\Pr}(x) \cdot \hat{\Pr}(y)}$
Gain ratio	$GR(t_k, c_j)$	$\frac{IG(t_k, c_j)}{-\sum_{x \in \{c_j, \overline{c}_j\}} \hat{\Pr}(x) \log \hat{\Pr}(x)}$
Pointwise MI	$PMI(t_k, c_j)$	$\log \frac{\hat{\Pr}(t_k, c_j)}{\hat{\Pr}(t_k) \cdot \hat{\Pr}(c_j)}$
Chi-square	$\chi^2(t_k, c_j)$	$\lvert Tr \rvert \cdot \frac{[\hat{\Pr}(t_k, c_j) \cdot \hat{\Pr}(\overline{t}_k, \overline{c}_j) - \hat{\Pr}(t_k, \overline{c}_j) \cdot \hat{\Pr}(\overline{t}_k, c_j)]^2}{\hat{\Pr}(t_k) \cdot \hat{\Pr}(\overline{t}_k) \cdot \hat{\Pr}(c_j) \cdot \hat{\Pr}(\overline{c}_j)}$
Odds Ratio	$OR(t_k, c_j)$	$\log \frac{\hat{\Pr}(t_k \lvert c_j) \cdot \hat{\Pr}(\overline{t}_k \lvert \overline{c}_j)}{\hat{\Pr}(\overline{t}_k \lvert c_j) \cdot \hat{\Pr}(t_k \lvert \overline{c}_j)}$

- Note that all these functions have a value of 0 when t_k and c_j, viewed as random variables, are independent.
- Note also that all these functions (except for PMI) reward not only the positive, but also the negative correlation of t_k to c_j.

- Example: Chi-square

$$\chi^2(t_k, c_j) \propto \frac{[\hat{\Pr}(t_k, c_j) \cdot \hat{\Pr}(\overline{t}_k, \overline{c}_j) - \hat{\Pr}(t_k, \overline{c}_j) \cdot \hat{\Pr}(\overline{t}_k, c_j)]^2}{\hat{\Pr}(t_k) \cdot \hat{\Pr}(\overline{t}_k) \cdot \hat{\Pr}(c_j) \cdot \hat{\Pr}(\overline{c}_j)}$$

  - In the experimental sciences χ^2 is used to measure how the results of an observation differ (i.e. are independent) from the results expected according to an initial hypothesis.
  - The terms t_k with the lowest value for $\chi^2(t_k, c_j)$ are thus the most independent from c_j; since we are interested in the terms which are not, we select the terms for which $\chi^2(t_k, c_j)$ is highest.

- All of the measures in Slide 82 (except *PMI*) tend to perform similarly well [4, 29]. Experiments with several classifiers and datasets [29] have shown that by using one of them for global DR
  - aggressivity 10 brings about a small effectiveness improvement;
  - aggressivity 100 brings about no effectiveness loss.

- For global DR (in the ML case) one usually picks the highest-scoring k terms for each c_j, in a "round robin" fashion, and takes the union of such sets [7].
- Note that we have presented these functions in a "binary form" (i.e. making reference to c_j and $\overline{c}_j$), which makes them suitable to the ML case. For the SL case they need to be modified accordingly.

- For global DR (in the ML case) one usually picks the highest-scoring k terms for each c_j, in a "round robin" fashion, and takes the union of such sets [7].
- Note that we have presented these functions in a "binary form" (i.e. making reference to c_j and $\overline{c}_j$), which makes them suitable to the ML case. For the SL case they need to be modified accordingly.

Outline

- The inductive construction of a classifier $\hat{\Phi}_j$ for category $c_j \in \mathcal{C}$ often consists of two different phases:
  1. identifying a function $CSV_j : \mathcal{D} \rightarrow [-1, 1]$ that, given d_i, returns a classification status value for it, representing the strength of the evidence for the fact that d_i belongs to c_j.
  2. identifying a threshold τ_j such that
     - $CSV_j(d_i) \geq \tau_j$ is interpreted as a decision to classify d_i under c_j;
     - $CSV_j(d_i) < \tau_j$ is interpreted as a decision not to classify d_i under c_j.
- A particular case occurs when the classifier already provides a binary judgment, i.e. $CSV_j : \mathcal{D} \rightarrow \{-1, +1\}$, in which case no threshold is needed.

- The inductive construction of a classifier $\hat{\Phi}_j$ for category $c_j \in \mathcal{C}$ often consists of two different phases:
  1. identifying a function $CSV_j : \mathcal{D} \rightarrow [-1, 1]$ that, given d_i, returns a classification status value for it, representing the strength of the evidence for the fact that d_i belongs to c_j.
  2. identifying a threshold τ_j such that
     - $CSV_j(d_i) \geq \tau_j$ is interpreted as a decision to classify d_i under c_j;
     - $CSV_j(d_i) < \tau_j$ is interpreted as a decision not to classify d_i under c_j.
- A particular case occurs when the classifier already provides a binary judgment, i.e. $CSV_j : \mathcal{D} \rightarrow \{-1, +1\}$, in which case no threshold is needed.

- The inductive construction of a classifier $\hat{\Phi}_j$ for category $c_j \in \mathcal{C}$ often consists of two different phases:
  1. identifying a function $CSV_j : \mathcal{D} \rightarrow [-1, 1]$ that, given d_i, returns a classification status value for it, representing the strength of the evidence for the fact that d_i belongs to c_j.
  2. identifying a threshold τ_j such that
     - $CSV_j(d_i) \geq \tau_j$ is interpreted as a decision to classify d_i under c_j;
     - $CSV_j(d_i) < \tau_j$ is interpreted as a decision not to classify d_i under c_j.
- A particular case occurs when the classifier already provides a binary judgment, i.e. $CSV_j : \mathcal{D} \rightarrow \{-1, +1\}$, in which case no threshold is needed.

- The most popular policies for determining the threshold τ_j on the values of CSV_j are [27]:
  - CSV thresholding : τ_j (which may or may not be equal for all c_j's) is obtained by optimization on a validation set.
  - proportional thresholding : τ_j is the value such that the validation set frequency $freq_{Va}(c_j)$ of c_j is as close as possible to its training set frequency $freq_{Tr}(c_j)$.
- CSV thresholding is theoretically better motivated, and produces superior effectiveness, but proportional thresholding is computationally cheaper, since one needs only to train a single classifier from Tr and apply it on Va.
- Thresholding is needed only
  - for "hard" classification, since in "soft" classification the final binary decision is taken by the expert, and the original non-binary CSV_j scores are used for ranking purposes;
  - for ML classification, as in SL classification we simply pick the category that has the highest CSV_j score.

- The most popular policies for determining the threshold τ_j on the values of CSV_j are [27]:
  - CSV thresholding : τ_j (which may or may not be equal for all c_j's) is obtained by optimization on a validation set.
  - proportional thresholding : τ_j is the value such that the validation set frequency $freq_{Va}(c_j)$ of c_j is as close as possible to its training set frequency $freq_{Tr}(c_j)$.
- CSV thresholding is theoretically better motivated, and produces superior effectiveness, but proportional thresholding is computationally cheaper, since one needs only to train a single classifier from Tr and apply it on Va.
- Thresholding is needed only
  - for "hard" classification, since in "soft" classification the final binary decision is taken by the expert, and the original non-binary CSV_j scores are used for ranking purposes;
  - for ML classification, as in SL classification we simply pick the category that has the highest CSV_j score.

- The most popular policies for determining the threshold τ_j on the values of CSV_j are [27]:
  - CSV thresholding : τ_j (which may or may not be equal for all c_j's) is obtained by optimization on a validation set.
  - proportional thresholding : τ_j is the value such that the validation set frequency $freq_{Va}(c_j)$ of c_j is as close as possible to its training set frequency $freq_{Tr}(c_j)$.
- CSV thresholding is theoretically better motivated, and produces superior effectiveness, but proportional thresholding is computationally cheaper, since one needs only to train a single classifier from Tr and apply it on Va.
- Thresholding is needed only
  - for "hard" classification, since in "soft" classification the final binary decision is taken by the expert, and the original non-binary CSV_j scores are used for ranking purposes;
  - for ML classification, as in SL classification we simply pick the category that has the highest CSV_j score.

- The most popular policies for determining the threshold τ_j on the values of CSV_j are [27]:
  - CSV thresholding : τ_j (which may or may not be equal for all c_j's) is obtained by optimization on a validation set.
  - proportional thresholding : τ_j is the value such that the validation set frequency $freq_{Va}(c_j)$ of c_j is as close as possible to its training set frequency $freq_{Tr}(c_j)$.
- CSV thresholding is theoretically better motivated, and produces superior effectiveness, but proportional thresholding is computationally cheaper, since one needs only to train a single classifier from Tr and apply it on Va.
- Thresholding is needed only
  - for "hard" classification, since in "soft" classification the final binary decision is taken by the expert, and the original non-binary CSV_j scores are used for ranking purposes;
  - for ML classification, as in SL classification we simply pick the category that has the highest CSV_j score.

- The most popular policies for determining the threshold τ_j on the values of CSV_j are [27]:
  - CSV thresholding : τ_j (which may or may not be equal for all c_j's) is obtained by optimization on a validation set.
  - proportional thresholding : τ_j is the value such that the validation set frequency $freq_{Va}(c_j)$ of c_j is as close as possible to its training set frequency $freq_{Tr}(c_j)$.
- CSV thresholding is theoretically better motivated, and produces superior effectiveness, but proportional thresholding is computationally cheaper, since one needs only to train a single classifier from Tr and apply it on Va.
- Thresholding is needed only
  - for "hard" classification, since in "soft" classification the final binary decision is taken by the expert, and the original non-binary CSV_j scores are used for ranking purposes;
  - for ML classification, as in SL classification we simply pick the category that has the highest CSV_j score.

Probabilistic classifiers

- Probabilistic classifiers view $CSV_j(d_i)$ in terms of $\Pr(c_j|\mathbf{d_i})$, where $\mathbf{d_i} = \langle w_{1i}, \ldots, w_{ni} \rangle$ is a vector of terms, and compute it by means of Bayes' theorem :

$$\Pr(c_j|\mathbf{d_i}) = \frac{\Pr(c_j)\Pr(\mathbf{d_i}|c_j)}{\Pr(\mathbf{d_i})} \quad (1)$$

- The estimation of $\Pr(\mathbf{d_i}|c_j)$ in Equation 1 is problematic. It is thus common to make the independence assumption

$$\Pr(\mathbf{d_i}|c_j) = \prod_{k=1}^{n} \Pr(w_{ki}|c_j) \quad (2)$$

- Probabilistic classifiers based on Equation 2 are called naïve Bayesian, and account for nearly all probabilistic approaches to TC.

Probabilistic classifiers

- Probabilistic classifiers view $CSV_j(d_i)$ in terms of $\Pr(c_j|\mathbf{d_i})$, where $\mathbf{d_i} = \langle w_{1i}, \ldots, w_{ni} \rangle$ is a vector of terms, and compute it by means of Bayes' theorem :

$$\Pr(c_j|\mathbf{d_i}) = \frac{\Pr(c_j)\Pr(\mathbf{d_i}|c_j)}{\Pr(\mathbf{d_i})} \quad (1)$$

- The estimation of $\Pr(\mathbf{d_i}|c_j)$ in Equation 1 is problematic. It is thus common to make the independence assumption

$$\Pr(\mathbf{d_i}|c_j) = \prod_{k=1}^{n} \Pr(w_{ki}|c_j) \quad (2)$$

- Probabilistic classifiers based on Equation 2 are called naïve Bayesian, and account for nearly all probabilistic approaches to TC.

Probabilistic classifiers

- Probabilistic classifiers view $CSV_j(d_i)$ in terms of $\Pr(c_j|\mathbf{d_i})$, where $\mathbf{d_i} = \langle w_{1i}, \ldots, w_{ni} \rangle$ is a vector of terms, and compute it by means of Bayes' theorem :

$$\Pr(c_j|\mathbf{d_i}) = \frac{\Pr(c_j)\Pr(\mathbf{d_i}|c_j)}{\Pr(\mathbf{d_i})} \quad (1)$$

- The estimation of $\Pr(\mathbf{d_i}|c_j)$ in Equation 1 is problematic. It is thus common to make the independence assumption

$$\Pr(\mathbf{d_i}|c_j) = \prod_{k=1}^{n} \Pr(w_{ki}|c_j) \quad (2)$$

- Probabilistic classifiers based on Equation 2 are called naïve Bayesian, and account for nearly all probabilistic approaches to TC.

- One of the best-known naïve Bayesian approaches is the binary independence classifier, which results from using binary weights. By applying Equations 1 and 2, with some algebra we obtain

$$\Pr(c_j|\mathbf{d_i}) \propto \sum_{k=1}^{n} w_{ki} \log \frac{\Pr(t_k|c_j)(1-\Pr(t_k|\overline{c}_j))}{\Pr(t_k|\overline{c}_j)(1-\Pr(t_k|c_j))}$$

- Note that all the factors for which $w_{ki} = 0$ may be disregarded, and this accounts for the vast majority of them, since document vectors are very sparse.
- Defining a classifier for c_j thus requires estimating the parameters $\Pr(t_k|c_j)$ and $\Pr(t_k|\overline{c}_j))$, for $i \in \{1, \ldots, n\}$. These are usually estimated by maximum likelihood (ML) with some form of smoothing; e.g. Laplace smoothing

$$\hat{\Pr}(t_k|c_j) = \frac{1 + |\{d_i \in Tr|\Phi_j(d_i) = 1 \ \& \ w_{ki} = 1\}|}{|Tr| + |\{d_i \in Tr|\Phi_j(d_i) = 1\}|} \quad (3)$$

- One of the best-known naïve Bayesian approaches is the binary independence classifier, which results from using binary weights. By applying Equations 1 and 2, with some algebra we obtain

$$\Pr(c_j|\mathbf{d_i}) \propto \sum_{k=1}^{n} w_{ki} \log \frac{\Pr(t_k|c_j)(1-\Pr(t_k|\overline{c}_j))}{\Pr(t_k|\overline{c}_j)(1-\Pr(t_k|c_j))}$$

- Note that all the factors for which $w_{ki} = 0$ may be disregarded, and this accounts for the vast majority of them, since document vectors are very sparse.
- Defining a classifier for c_j thus requires estimating the parameters $\Pr(t_k|c_j)$ and $\Pr(t_k|\overline{c}_j))$, for $i \in \{1, \ldots, n\}$. These are usually estimated by maximum likelihood (ML) with some form of smoothing; e.g. Laplace smoothing

$$\hat{\Pr}(t_k|c_j) = \frac{1 + |\{d_i \in Tr|\Phi_j(d_i) = 1 \ \& \ w_{ki} = 1\}|}{|Tr| + |\{d_i \in Tr|\Phi_j(d_i) = 1\}|} \quad (3)$$

- One of the best-known naïve Bayesian approaches is the binary independence classifier, which results from using binary weights. By applying Equations 1 and 2, with some algebra we obtain

$$\Pr(c_j|\mathbf{d_i}) \propto \sum_{k=1}^{n} w_{ki} \log \frac{\Pr(t_k|c_j)(1-\Pr(t_k|\overline{c}_j))}{\Pr(t_k|\overline{c}_j)(1-\Pr(t_k|c_j))}$$

- Note that all the factors for which $w_{ki} = 0$ may be disregarded, and this accounts for the vast majority of them, since document vectors are very sparse.
- Defining a classifier for c_j thus requires estimating the parameters $\Pr(t_k|c_j)$ and $\Pr(t_k|\overline{c}_j))$, for $i \in \{1, \ldots, n\}$. These are usually estimated by maximum likelihood (ML) with some form of smoothing; e.g. Laplace smoothing

$$\hat{\Pr}(t_k|c_j) = \frac{1 + |\{d_i \in Tr|\Phi_j(d_i) = 1 \ \& \ w_{ki} = 1\}|}{|Tr| + |\{d_i \in Tr|\Phi_j(d_i) = 1\}|} \qquad (3)$$

Learning linear classifiers

- A linear classifier is a classifier such that classification is performed by a dot product between the two vectors representing the document and the category, respectively. Therefore, it consists in a document-like representation of the category c_j.
- Linear classifiers are thus very efficient at classification time.
- Methods for learning linear classifiers can be partitioned in two broad classes:
  - incremental methods (IMs) build a classifier soon after examining the first training document, and incrementally refine it as they examine new ones.
  - batch methods (BMs) build a classifier by analysing Tr all at once. The foremost example of a BM is the Rocchio method.

Learning linear classifiers
Incremental methods

A simple incremental, "additive weight-updating" method is the perceptron algorithm:

1. Initialize $\hat{\Phi}_j$ by setting all weights w_{ki} to the same positive value a.
2. For all training examples d_i do
   1. Classify d_i by $\hat{\Phi}_j$.
   2. If $\hat{\Phi}_j(d_i) \neq \Phi_j(d_i)$ modify the weights w_{kj} of all terms t_k such that $w_{ki} = 1$:
      - if d_i is a positive example of c_j then increase w_{ki} by the fixed quantity $\alpha > 0$ (learning rate);
      - if d_i is a negative example of c_j then decrease w_{ki} by α.

Example-based classifiers

- Example-based classifiers (EBCs) learn from the categories of the training documents similar to the one to be classified.
- The most frequently used EBC is the distance-weighted k-NN [25], which comes down to computing

$$CSV_j(d_i) = \frac{1}{k} \sum_{d_z \in Tr_k(d_i)} RSV(d_i, d_z) \cdot \Phi_j(d_z) \qquad (4)$$

where

  - $RSV(d_i, d_z)$ represents a measure of semantic relatedness between d_i and d_z. Any matching function, be it probabilistic or vector-based, may be used for this purpose.
  - $Tr_k(d_i)$ is the set of the k documents d_z for which $RSV(d_i, d_z)$ is maximum;
  - the $\frac{1}{k}$ factor is a normalization constant, such that $-1 \leq CSV_j(d_i) \leq 1$.

- For building a k-NN classifier we also need to optimize the value of k on a validation set; typical values range from $k = 20$ [14] to $30 \leq k \leq 45$ [26, 28].
- Drawbacks:
  - lazy learners such as k-NN are less efficient than eager methods at classification time, as they defer all the computation to classification time.
  - Quite evidently, EBCs are only applicable to DPC; in CPC they would be unacceptably inefficient.
- Advantages:
  - k-NN, unlike linear classifiers, does not subdivide the document space in just two subspaces, hence it does not suffer from the "linear separation problem". k-NN has been shown to be among the best performers in [10, 26, 28].
  - k-NN becomes comparatively efficient for large category sets, since ranking the training documents is done only once, for all categories [30].

Classifiers based on support vector machines

- The support vector machine (SVM) method attempts to find, among all the decision surfaces $\hat{\Phi}_1, \hat{\Phi}_2, \ldots$ in n-dimensional space, the one $\hat{\Phi}$ that does it by the widest possible margin.
- This method applies the so-called structural risk minimization principle, according to which the decision surface should minimize true error, i.e. the probability of misclassification of a randomly selected, yet unseen test example.
- If the positives and the negatives are linearly separable, the decision surfaces are hyperplanes. The SVM chooses the widest set of parallel hyperplanes (and from this its middle element), i.e. the one in which the maximum distance between two elements in the set is highest.
- Learning a SVM classifier is a quadratic programming problem.

Classifiers based on support vector machines

- The support vector machine (SVM) method attempts to find, among all the decision surfaces $\hat{\Phi}_1, \hat{\Phi}_2, \ldots$ in n-dimensional space, the one $\hat{\Phi}$ that does it by the widest possible margin.
- This method applies the so-called structural risk minimization principle, according to which the decision surface should minimize true error, i.e. the probability of misclassification of a randomly selected, yet unseen test example.
- If the positives and the negatives are linearly separable, the decision surfaces are hyperplanes. The SVM chooses the widest set of parallel hyperplanes (and from this its middle element), i.e. the one in which the maximum distance between two elements in the set is highest.
- Learning a SVM classifier is a quadratic programming problem.

Classifiers based on support vector machines

- The support vector machine (SVM) method attempts to find, among all the decision surfaces $\hat{\Phi}_1, \hat{\Phi}_2, \ldots$ in n-dimensional space, the one $\hat{\Phi}$ that does it by the widest possible margin.
- This method applies the so-called structural risk minimization principle, according to which the decision surface should minimize true error, i.e. the probability of misclassification of a randomly selected, yet unseen test example.
- If the positives and the negatives are linearly separable, the decision surfaces are hyperplanes. The SVM chooses the widest set of parallel hyperplanes (and from this its middle element), i.e. the one in which the maximum distance between two elements in the set is highest.
- Learning a SVM classifier is a quadratic programming problem.

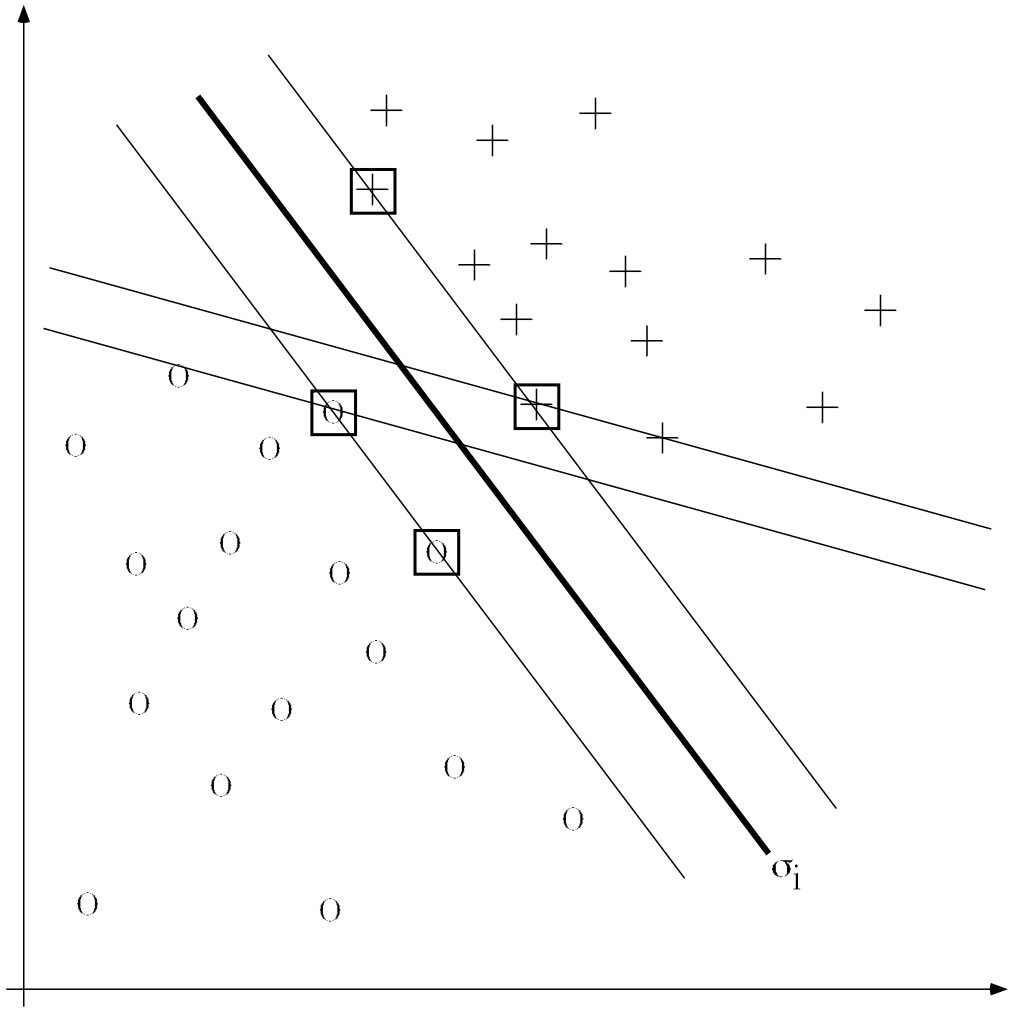

Figure: The hyperplane generated by a linear SVM.

- SVMs have important advantages for TC:
  - The "best" decision surface is determined by only a small set of training examples, called the support vectors.
  - Different "kernel" functions can be plugged in, corresponding to different ways of computing the similarity of two documents.
  - The method is applicable also to the case in which the positives and the negatives are not linearly separable.
  - No term selection is usually needed, as SVMs are fairly robust to overfitting and can scale up to high dimensionalities.
- SVMs have been shown among the top-performing systems in a number of experiments [5, 10, 28].
- There are several public-domain implementations of SVMs such as SVMlight and libSVM.

Bibliography


S. Baccianella, A. Esuli, and F. Sebastiani.
Multi-facet rating of product reviews.
In *Proceedings of the 31st European Conference on Information Retrieval (ECIR'09)*, pages 461–472, Toulouse, FR, 2009.


M. F. Caropreso, S. Matwin, and F. Sebastiani.
A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization.
In A. G. Chin, editor, *Text Databases and Document Management: Theory and Practice*, pages 78–102. Idea Group Publishing, Hershey, US, 2001.


K. Crammer and Y. Singer.
Pranking with ranking.
In *Advances in Neural Information Processing Systems*, volume 14, pages 641–647. The MIT Press, Cambridge, US, 2002.


F. Debole and F. Sebastiani.
Supervised term weighting for automated text categorization.
In *Proceedings of the 18th ACM Symposium on Applied Computing (SAC'03)*, pages 784–788, Melbourne, US, 2003.


S. T. Dumais, J. Platt, D. Heckerman, and M. Sahami.
Inductive learning algorithms and representations for text categorization.
In *Proceedings of the 7th ACM International Conference on Information and Knowledge Management (CIKM'98)*, pages 148–155, Bethesda, US, 1998.


A. Finn, N. Kushmerick, and B. Smyth.
Genre classification and domain transfer for information filtering.
In *Proceedings of the 24th European Colloquium on Information Retrieval Research (ECIR'02)*, pages 353–362, Glasgow, UK, 2002.


G. Forman.
A pitfall and solution in multi-class feature selection for text classification.
In *Proceedings of the 21st International Conference on Machine Learning (ICML'04)*, pages 38–45, Banff, CA, 2004.


D. Giorgetti and F. Sebastiani.
Automating survey coding by multiclass text categorization techniques.
Journal of the American Society for Information Science and Technology, 54(14):1269–1277, 2003.


T. Gottron and N. Lipka.
A comparison of language identification approaches on short, query-style texts.
In *Proceedings of the 32nd European Conference on Information Retrieval (ECIR'10)*, pages 611–614, Milton Keynes, UK, 2010.


T. Joachims.
Text categorization with support vector machines: Learning with many relevant features.
In *Proceedings of the 10th European Conference on Machine Learning (ECML'98)*, pages 137–142, Chemnitz, DE, 1998.


P. Juola.
Authorship attribution.
Foundations and Trends in Information Retrieval, 1(3):233—334, 2006.


M. Koppel, S. Argamon, and A. R. Shimoni.
Automatically categorizing written texts by author gender.

Literary and Linguistic Computing, 17(4):401–412, 2002.

C. H. Koster and M. Seutter.

Taming wild phrases.

In *Proceedings of the 25th European Conference on Information Retrieval (ECIR'03)*, pages 161–176, Pisa, IT, 2003.

L. S. Larkey and W. B. Croft.

Combining classifiers in text categorization.

In *Proceedings of the 19th ACM International Conference on Research and Development in Information Retrieval (SIGIR'96)*, pages 289–297, Zürich, CH, 1996.

Y.-B. Lee and S. H. Myaeng.

Text genre classification with genre-revealing and subject-revealing features.

In *Proceedings of the 25th ACM International Conference on Research and Development in Information Retrieval (SIGIR'02)*, pages 145–150, Tampere, FI, 2002.

D. D. Lewis.

Representation and learning in information retrieval.

PhD thesis, Department of Computer Science, University of Massachusetts, Amherst, US, 1992.

T. Macer, M. Pearson, and F. Sebastiani.

Cracking the code: What customers say, in their own words.

In *Proceedings of the 50th Annual Conference of the Market Research Society (MRS'07)*, Brighton, UK, 2007.

M. Maron.

Automatic indexing: an experimental inquiry.

Journal of the Association for Computing Machinery, 8(3):404–417, 1961.


S. Meyer Zu Eissen and B. Stein.
Genre classification of Web pages.
In *Proceedings of the 27th German Conference on Artificial Intelligence (KI'04)*, pages 256–269, Ulm, DE, 2004.


K. Myers, M. Kearns, S. Singh, and M. A. Walker.
A boosting approach to topic spotting on subdialogues.
In *Proceedings of the 17th International Conference on Machine Learning (ICML'00)*, pages 655–662, Stanford, US, 2000.


B. Pang, L. Lee, and S. Vaithyanathan.
Thumbs up? Sentiment classification using machine learning techniques.
In *Proceedings of the 7th Conference on Empirical Methods in Natural Language Processing (EMNLP'02)*, pages 79–86, Philadelphia, US, 2002.


C. L. Sable and V. Hatzivassiloglou.
Text-based approaches for non-topical image categorization.
International Journal of Digital Libraries, 3(3):261–275, 2000.


R. E. Schapire and Y. Singer.
Boostexter: A boosting-based system for text categorization.
Machine Learning, 39(2/3):135–168, 2000.


F. Sebastiani.
Machine learning in automated text categorization.
ACM Computing Surveys, 34(1):1–47, 2002.


Y. Yang.
Expert network: Effective and efficient learning from human decisions in text categorisation and retrieval.

In *Proceedings of the 17th ACM International Conference on Research and Development in Information Retrieval (SIGIR'94)*, pages 13–22, Dublin, IE, 1994.

Y. Yang.
An evaluation of statistical approaches to text categorization.
Information Retrieval, 1(1/2):69–90, 1999.

Y. Yang.
A study on thresholding strategies for text categorization.
In *Proceedings of the 24th ACM International Conference on Research and Development in Information Retrieval (SIGIR'01)*, pages 137–145, New Orleans, US, 2001.

Y. Yang and X. Liu.
A re-examination of text categorization methods.
In *Proceedings of the 22nd ACM International Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 42–49, Berkeley, US, 1999.

Y. Yang and J. O. Pedersen.
A comparative study on feature selection in text categorization.
In *Proceedings of the 14th International Conference on Machine Learning (ICML'97)*, pages 412–420, Nashville, US, 1997.

Y. Yang, J. Zhang, and B. Kisiel.
A scalability analysis of classifiers in text categorization.
In *Proceedings of the 26th ACM International Conference on Research and Development in Information Retrieval (SIGIR'03)*, pages 96–103, Toronto, CA, 2003.