I put my heart and my soul into my work, and have lost my mind in the process.

Vincent van Gogh

2 Text Analysis Pipelines

The understanding of natural language is one of the primary abilities that provide the basis for human intelligence. Since the invention of computers, people have thought about how to operationalize this ability in software applications (Jurafsky and Martin, 2009). The rise of the internet in the 1990's then made explicit the practical need for automatically processing natural language in order to access relevant information. Search engines, as a solution, have revolutionalized the way we can find such information ad-hoc in large amounts of text (Manning et al., 2008). Until today, however, search engines excel in finding relevant texts rather than in understanding what information is relevant in the texts. Chapter 1 has proposed text mining as a means to achieve progress towards the latter, thereby making information search more intelligent. At the heart of every text mining application lies the analysis of text, mostly realized in the form of text analysis pipelines. In this chapter, we present the basics required to follow the approaches of this thesis to improve such pipelines for enabling text mining ad-hoc on large amounts of text as well as the state of the art in this respect.

Text mining combines techniques from information retrieval, natural language processing, and data mining. In Section 2.1, we first provide a focused overview of those techniques referred to in this thesis. Then, we define the text analysis processes and pipelines that we consider in our proposed approaches (cf. Section 2.2). We evaluate the different approaches based on texts and pipelines from a number of case studies introduced in Section 2.3. Finally, Section 2.4 surveys and discusses related existing work in the broad context of ad-hoc large-scale text mining.

2.1 Foundations of Text Mining

In this section, we explain all general foundations of text mining the thesis at hand builds upon. After a brief outline of text mining, we organize the foundations along the three main research fields related to text mining. The goal is not to provide a formal and comprehensive introduction into these fields, but rather to give exactly the information that is necessary to follow our discussion. At the end, we describe how to develop and evaluate approaches to text analysis. Basic concepts of text analysis processes are defined in Section 2.2, while specific concepts related to our overall approach are directly defined where needed in Chapters 3 to 5.[1]

Text Mining

Text mining deals with the automatic or semi-automatic discovery of new, previously unknown information of high quality from large numbers of unstructured texts (Hearst, 1999). Different than sometimes assumed, the types of information to be inferred from the texts are usually specified manually beforehand, i.e., text mining tackles given tasks. As introduced in Section 1.1, this commonly requires to perform three steps in sequence, each of which can be associated to one field (Ananiadou and McNaught, 2005):

1. **Information retrieval.** Gather input texts that are potentially relevant for the given task.
2. **Natural language processing.** Analyze the input texts in order identify and structure relevant information.[2]
3. **Data mining.** Discover patterns in the structured information that has been inferred from the texts.

Hearst (1999) points out that the main aspects of text mining are actually the same as those studied in empirical computational linguistics. Although focusing on natural language processing, some of the problems computational linguistics is concerned with are also addressed in information retrieval and data mining, such as text classification or machine learning. In this thesis, we refer to all these aspects with the general term text analysis (cf. Section 1.1). In the following, we look at the concepts of the three fields that are important for our discussion of text analysis.

[1] Notice that, throughout this thesis, we generally assume that the reader has a more or less graduate-level background in computer science.

[2] Ananiadou and McNaught (2005) refer to the second step as information extraction. While we agree that information extraction is often the important part of this step, also other techniques from natural language processing play a role, as discussed later in this section.

Information Retrieval

Following Manning et al. (2008), the primary use case of information retrieval is to search and obtain those texts from a large collection of unstructured texts that can satisfy an information need, usually given in the form of a query. In ad-hoc web search, such a query consists of a few keywords, but, in general, it may also be given by a whole text, a logical expression, etc. An information retrieval application assesses the relevance of all texts with respect to a query based on some similarity measure. Afterwards, it ranks the texts by decreasing relevance or it filters only those texts that are classified as potentially relevant (Manning et al., 2008).

Although the improvement of ad-hoc search denotes one of the main motivations behind this thesis (cf. Chapter 1), we hardly consider the retrieval step of text mining, since we focus on the inference of information from the potentially relevant texts, as we detail in Section 2.2. Still, we borrow some techniques from information retrieval, such as filtering or the determination of similar texts. For this purpose, we require the following concepts, which are associated to information retrieval rather than to text analysis.

Vectors To determine the relevance of texts, many approaches map all texts and queries into a *vector space model* (Manning et al., 2008). In general, such a model defines a common vector representation $\mathbf{x} = (x_1, \ldots, x_k)$, $k \geq 1$, for all inputs, where each $x_i \in \mathbf{x}$ formalizes an input property. A concrete input like a text D is then represented by one value $x_i^{(D)}$ for each x_i. In web search, the standard way to represent texts and queries is by the frequencies of the words they contain from a set of (possibly hundreds of thousands) words. Generally, any measurable property of an input can be formalized, though, which becomes particularly relevant for tasks like text classification.

vector space model

Similarity Given a common representation, similarities between texts and queries can be computed. Most word frequencies of a search query will often be 0. In case they are of interest, a reasonable similarity measure is the cosine distance, which puts emphasis on the properties that occur (Manning et al., 2008). In Chapter 5, we compute similarities of whole texts, where a zero does not always mean the absence of a property. Such scenarios suggest other measures. In our experiments, we use the *Manhattan distance* between two vectors $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ of length k (Cha, 2007), which is defined as:

manhattan distance

$$\text{Manhattan distance}(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \sum_{i=1}^{k} |\mathbf{x}_i^{(1)} - \mathbf{x}_i^{(2)}|$$

Indexing While queries are typically stated ad-hoc, the key to efficient ad-hoc search is that all texts in a given collection have been indexed before.

search index A query is then matched against the *search index*, thereby avoiding to process the actual texts during search. Very sophisticated indexing approaches exist and are used in today's web search engines (Manning et al., 2008). In its basic form, a search index contains one entry for every measured property. Each entry points to all texts that are relevant with respect to the property. Some researchers have adapted indexing to information extraction by building specialized search indexes based on concepts like entities, such as Cafarella et al. (2005). We discuss in Section 2.4 in how far they reduce the need for ad-hoc large-scale text mining that we address in this thesis.

Filtering While the ranking of texts by relevance is not needed in this thesis, we filter relevant portions of texts in Chapter 3. Filtering is addressed in information retrieval on two levels: *Text filtering* classifies complete texts as being relevant or irrelevant (Sebastiani, 2002), whereas *passage retrieval* aims to determine the passages of a text that are relevant for answering a given query (Cui et al., 2005). We investigate the difference between our and existing filtering approaches in Section 2.4 and their integration at the end of Chapter 3. Filtering is usually seen as a classification task (Manning et al., 2008) and, thus, addressed as a text analysis. We cover text classification as part of natural language processing, which we describe next.

text filtering

passage retrieval

Natural Language Processing

Natural language processing covers algorithms and engineering issues for the understanding and generation of speech and human-readable text (Tsujii, 2011). In the thesis at hand, we concentrate on the analysis of text with the goal of deriving structured information from unstructured texts.

In text analysis, algorithms are employed that, among others, infer lexical information about the words in a text, syntactic information about the structure between words, and semantic information about the meaning of words (Manning and Schütze, 1999). Also, they may analyze the discourse and pragmatic level of a text (Jurafsky and Martin, 2009). In Chapters 3 to 5, we use lexical and syntactic analyses as preprocessing for information extraction and text classification. Information extraction targets at semantic information. Text classification may seek for both semantic and pragmatic information. To infer information of certain types from an input text, text analysis algorithms apply rules or statistics, as we detail below.

ambiguity Generally, natural language processing faces the problem of *ambiguity*, i.e., many utterances of natural language allow for different interpretations. As a consequence, all text analysis algorithms need to resolve ambiguities (Jurafsky and Martin, 2009). Without sufficient context, a correct ana-

lysis is hence often hard and can even be impossible. For instance, the sentence *"SHE'S AN APPLE FAN."* alone leaves undecidable whether it refers to a fruit or a company.

Technically, natural language processing can be seen as the production of *annotations* (Ferrucci and Lally, 2004). An annotation marks a text or a span of text that represents an instance of a particular type of information. We discuss the role of annotations more extensively in Section 2.2, before we formalize the view of text analysis as an annotation task in Chapter 3.

ANNOTATION

Lexical and Syntactic Analyses For our purposes, we distinguish three types of lexical and syntactical analyses: The *segmentation* of a text into single units, the *tagging* of units, and the *parsing* of syntactic structure.

SEGMENTATION
TAGGING
PARSING

Mostly, the smallest text unit considered in natural language processing is a *token*, denoting a word, a number, a symbol, or anything similar (Manning and Schütze, 1999). Besides the *tokenization* of texts, we also perform segmentation with *sentence splitting* and *paragraph splitting* in this thesis. In terms of tagging, we look at *part-of-speech*, meaning the categories of tokens like nouns or verbs, although much more specific *part-of-speech tags* are used in practice (Jurafsky and Martin, 2009). Also, we perform *lemmatization* in some experiments to get the *lemmas* of tokens, i.e., their dictionary forms, such as "be" in case of "is" (Manning and Schütze, 1999). Finally, we use shallow parsing, called *chunking* (Jurafsky and Martin, 2009), to identify different types of phrases, and *dependency parsing* to infer the dependency tree structure of sentences (Bohnet, 2010). Appendix A provides details on all named analyses and on the respective algorithms we rely on. The output of parsing is particularly important for information extraction.

TOKEN
TOKENIZATION
SENTENCE SPLITTING
PARAGRAPH SPLITTING
PART-OF-SPEECH
PART-OF-SPEECH TAG
LEMMATIZATION
LEMMA
CHUNKING
DEPENDENCY PARSING

Information Extraction The basic semantic concept is a named or numeric *entity* from the real world (Jurafsky and Martin, 2009). Information extraction analyzes usually unstructured texts in order to recognize references of such entities, *relations* between entities, and *events* the entities participate in (Sarawagi, 2008). In the classical view of the Message Understanding Conferences, information extraction is seen as a template filling task (Chinchor et al., 1993), where the goal is to fill entity slots of relation or event templates with information from a collection or a stream of texts $\mathbf{D}$.

ENTITY
RELATION
EVENT

The set of information types $\mathbf{C}$ to be recognized is often predefined, although some recent approaches address this limitation (cf. Section 2.4). Both rule-based approaches, e.g. based on regular expressions or lexicons, and statistical approaches, mostly based on machine learning (see below), are applied in information extraction (Sarawagi, 2008). The output is structured information that can be stored in databases or directly displayed to the

users (Cunningham, 2006). As a matter of fact, information extraction plays an important role in today's database research (Chiticariu et al., 2010a), while it has its origin in computational linguistics (Sarawagi, 2008). In principle, the output qualifies for being exploited in text mining applications, e.g. to provide relevant information like Google in the example from Figure 1.2 (Section 1.1). However, many information types tend to be domain-specific and application-specific (Cunningham, 2006), making their extraction cost-intensive. Moreover, while some types can be extracted accurately, at least from hiqh-quality texts of common languages (Ratinov and Roth, 2009), others still denote open challenges in current research (Ng, 2010).

COREFERENCE RESOLUTION
NORMALIZATION
ENTITY RECOGNITION
RELATION EXTRACTION
EVENT DETECTION

Information extraction often involves a number of subtasks, including *coreference resolution*, i.e., the identification of references that refer to the same entity (Cunningham, 2006), and the *normalization* of entities and the like. In this thesis, we focus mainly on the most central subtasks, namely, named and numeric *entity recognition*, binary *relation extraction*, and *event detection* (Jurafsky and Martin, 2009). Concrete analyses as well as algorithms that realize the analyses are found in Appendix A. As an example, Figure 2.1(a) illustrates instances of different information types in a sample text. Some refer to a relation of the type *Founded(Organization, Time)*.

In Section 2.3, we introduce the more sophisticated extraction of financial events from news articles that we consider in many experiments in this thesis. Without exception, we extraction information only *within* but not *across* texts unlike e.g. Li et al. (2011). Also, we restrict our view to unstructured texts and, hence, omit to present approaches that target at structured or semi-structured texts like wrapper induction (Kushmerick, 1997).

Text Classification Text classification (or text categorization) denotes the task of assigning each text from a collection or a stream of texts $\mathbf{D}$ to one of a set of predefined *classes* $C = \{c_1, \ldots, c_k\}$, $k \geq 2$ (Jurafsky and Martin, 2009). We call C the *classification scheme* here. The standard approach to text classification is to statistically learn an assignment based on a set of training texts with known classes (Manning et al., 2008). To this end, every text is converted into a vector representation consisting of a number of (typically lexical or shallow syntactic) features of the text (Sebastiani, 2002). We introduce this representation as part of the data mining foundations below.

CLASS
CLASSIFICATION SCHEME

Several text classification tasks are studied in natural language processing (Manning and Schütze, 1999) and information retrieval (Manning et al., 2008). The classic *topic detection* (Lewis et al., 2004) targets at the main topic of a text, whereas in *non-standard text classification tasks*, classes go beyond the subjects of texts (Lipka, 2013). Examples are the identification of the

TOPIC DETECTION
NON-STANDARD TEXT CLASSIFICATION TASK

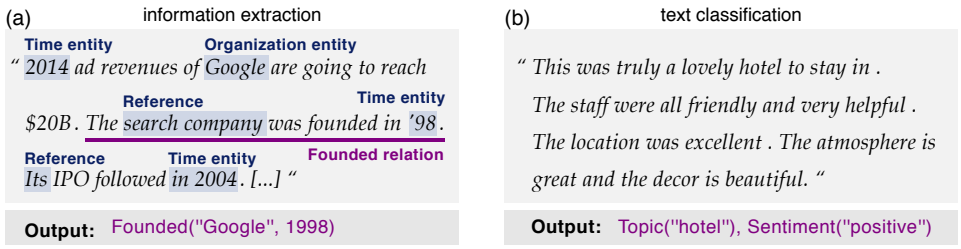

Figure 2.1: (a) An information extraction example: Extraction of a relation of the type *Founded(Organization, Time)* from a sample text. (b) A text classification example: Classification of the topic and the sentiment polarity of a sample text.

genre of a text in terms of the form, purpose, and/or intended audience of the text (Stein et al., 2010), or *authorship attribution* where the author of a text is to be determined (Stamatatos, 2009). In *automatic essay grading*, the goal is to assign ratings from a usually numeric classification scheme to texts like essays (Dikli, 2006), and *stance recognition* seeks for the stance of a person with respect to some topic (Somasundaran and Wiebe, 2010). All these and some related tasks are discussed more or less detailed in this thesis.

genre
authorship attribution
automatic essay grading
stance recognition

Of highest importance in our experiments is *sentiment analysis*, which has become one of the most widely investigated text classification tasks in the last decade. By default, sentiment analysis refers to the classification of the *sentiment polarity* of a text as being positive or negative (Pang et al., 2002). An example is shown in Figure 2.1(b). Sometimes, also an objective (or neutral) "polarity" is considered, although this class rather refers to *subjectivity* (Pang and Lee, 2004). Moreover, sentiment can also be assessed on numeric scales (Pang and Lee, 2005), which we call *sentiment scoring* here. We employ a number of sentiment analysis algorithms in Section 5. They are listed in Appendix A.

sentiment analysis
sentiment polarity
subjectivity
sentiment scoring

As passage retrieval (see above), text classification does not always deal with complete texts. In Chapter 5, we classify the subjectivity of single discourse units, where objective units can be seen as *facts* and subjective units as *opinions*. In *opinion mining*, such techniques are combined with information extraction techniques to find opinions on certain topics (Popescu and Etzioni, 2005), as done in one of our case studies (cf. Section 2.3).[3] Sentiment analysis and opinion mining are of high practical relevance, because they can be used in text mining applications that analyze the people's opinions on products and brands in social media, online review sites, and the like (Pang and Lee, 2008). For this purpose, data mining needs to be performed on the output of the respective algorithms.

fact
opinion
opinion mining

[3] Unlike us, some researchers do not distinguish between sentiment analysis and opinion mining, but they use these two terms interchangeably (Pang and Lee, 2008).

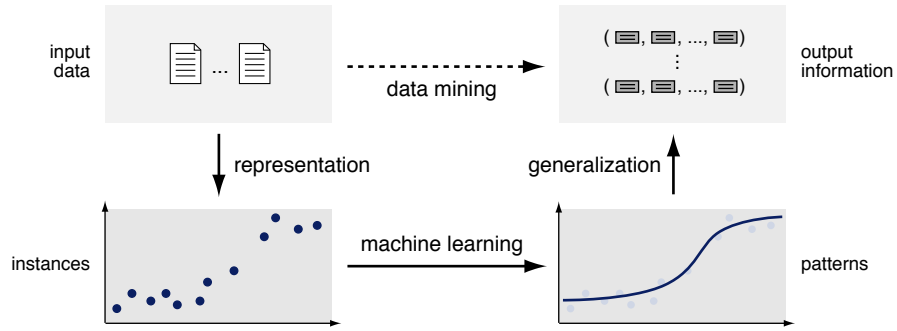

Figure 2.2: Illustration of a high-level view of data mining. Input data is represented as a set of instances, from which a model is derived using machine learning. The model is then generalized to infer new output information.

Data Mining

Data mining primarily aims at the inference of new information of specified types from typically huge amounts of input data, already given in structured form (Witten and Frank, 2005). To address such a *prediction problem*, the data is first converted into instances of a defined representation and then handed over to a *machine learning* algorithm. The algorithm recognizes statistical patterns in the instances that are relevant for the prediction problem. This process is called *training*. The found patterns are then generalized, such that they can be applied to infer new information from unseen data, generally referred to as *prediction*. In this regard, machine learning can be seen as the technical basis of data mining applications (Witten and Frank, 2005). Figure 2.2 shows a high-level view of the outlined process.

PREDICTION PROBLEM
MACHINE LEARNING
TRAINING
PREDICTION

Data mining and text mining are related in two respects: On one hand, the structured output information of text analysis serves as the input to machine learning, e.g. to train a text classifier. On the other hand, many text analyses themselves rely on machine learning algorithms to produce output information. Both respects are important in this thesis. In the following, we summarize the basic concepts relevant for our purposes.[4]

Machine Learning Machine learning describes the ability of an algorithm to learn without being explicitly programmed (Samuel, 1959). An algorithm can be said to learn from data with respect to a given prediction problem and some quality measure, if the measured prediction quality increases the more data is processed (Mitchell, 1997).[5] Machine learning aims at prediction problems where the *target function*, which maps input data to output in-

TARGET FUNCTION

[4] Besides the cited references, parts of the summary are inspired by the Coursera machine learning course, `https://www.coursera.org/course/ml` (accessed on December 2, 2014).

[5] A discussion of common quality measures follows at the end of this section.

formation, is unknown and which, thus, cannot be (fully) solved by following hand-crafted rules. In the end, all non-trivial text analysis tasks denote such prediction problems, even though many tasks have been successfully tackled with rule-based approaches.[6]

A machine learning algorithm produces a *model* $\mathcal{Y} : \mathbf{x} \rightarrow C$, which generalizes patterns found in the input data in order to approximate the target function. $\mathcal{Y}$ defines a mapping from represented data $\mathbf{x}$ to a *target variable* C, where C captures the type of information sought for. In text analysis, the target variable may represent classes of texts (e.g. topics or genres), types of annotations (e.g. part-of-speech tags or entity types), etc. Since machine learning generalizes from examples, the learned prediction of output information cannot be expected to be correct in all cases. Rather, the goal is to find a model $\mathcal{Y}$ that is optimal with respect to a given quality measure (see below). Besides the input data, the quality of $\mathcal{Y}$ depends on how the data is represented and how the found patterns are generalized.

MODEL

TARGET VARIABLE

Representation Similar to information retrieval, most machine learning algorithms rely on a vector space model. In particular, the input data is represented by a set X of *feature vectors* of the form $\mathbf{x}$. $\mathbf{x}$ defines an ordered set of *features*, where each feature $x \in \mathbf{x}$ denotes a measurable property of an input (Hastie et al., 2009). In text mining, common features are e.g. the frequency of a particular word in a given text or the shape of a word (say, capitalized or not). Representing input data means to create a set of instances of $\mathbf{x}$, such that each instance contains one *feature value* for every feature in $\mathbf{x}$.[7] In many cases, hundreds or thousands of features are considered in combination. They belong to different *feature types*, like *bag-of-words* where each feature means the frequency of a word (Manning et al., 2008).

FEATURE VECTOR

FEATURE

FEATURE VALUE

FEATURE TYPE

BAG-OF-WORDS

The feature representation of the input data governs what patterns can be found during learning. As a consequence, the development of features, which predict a given target variable C, is one of the most important (and often most difficult) steps in machine learning. Although common feature types like bag-of-words help in many text analysis tasks, the most discriminative features tend to require expert knowledge about the task and input.[8]

[6] The question for what text analysis tasks to prefer a rule-based approach over a machine learning approach is out of the scope of this thesis.

[7] Throughout this thesis, we consider only features whose values come from a metric scale. Other features are transformed, e.g. a feature with values "red", "green", and "blue" can be represented by three 0/1-features, one for each value. All values are normalized to the same interval, namely [0,1], which benefits learning (Witten and Frank, 2005).

[8] The concrete features of a feature type can often be chosen automatically based on input data, as we do in our experiments, e.g. by taking only those words whose occurrence is above some threshold. Thereby, useless features that would introduce noise are excluded.

Also, some features generalize worse than others, often because they capture domain-specific properties, as we see in Chapter 5.[9]

Generalization As shown in Figure 2.2, generalization refers to the inference of output information from unseen data based on patterns captured in a learned model (Witten and Frank, 2005). As such, it is strongly connected to the used machine learning algorithm. The training of such an algorithm based on a given set of instances explores a large space of models, because most algorithms have a number of parameters. An important decision in this regard is how much to bias the algorithm with respect to the complexity of the model to be learned (Witten and Frank, 2005). Simple models (say, linear functions) induce a high bias, which may not fit the input data well, but regularize noise in the data and, thus, tend to generalize well. Complex models (say, high polynomials) can be fitted well to the data, but tend to generalize less. We come back to this problem of *fitting* in Section 5.1.

FITTING

During training, a machine learning algorithm incrementally chooses a possible model and evaluates the model based on some cost function. The choice relies on an optimization procedure, e.g. *gradient descent* stepwise heads towards a local minimum of the cost function until convergence by adapting the model to all input data (Witten and Frank, 2005). In large-scale scenarios, a variant called *stochastic gradient descent* is often more suitable. It repeatedly iterates over all data instances in isolation, thereby being much faster while not guaranteeing to find a local minimum (Zhang, 2004). No deep understanding of the generalization process is needed in this thesis, since we focus only on the question how to address text analysis tasks with existing machine learning algorithms in order to then select an adequate one. What matters for us is the type of learning that can or should be performed within the task at hand. Mainly, we consider two very prominent types in this thesis, supervised learning and unsupervised learning.

GRADIENT DESCENT

STOCHASTIC GRADIENT DESCENT

Supervised Learning In *supervised learning*, a machine learning algorithm derives a model from known *training data*, i.e., pairs of a data instance and the associated output information (Witten and Frank, 2005). The model can then be used to predict output information for unknown data. The notion of being supervised refers to the fact that the learning process is guided by examples of correct predictions. In this thesis, we use supervised learning for both statistical classification and statistical regression.

SUPERVISED LEARNING

TRAINING DATA

Classification describes the task to assign a data instance to the most likely of a set of two or more predefined discrete classes (Witten and Frank, 2005).

CLASSIFICATION

[9] Techniques like feature selection and dimensionality reduction, which aim to reduce the set of considered features to improve generalizability and training efficiency among others (Hastie et al., 2009), are beyond the scope of this thesis.

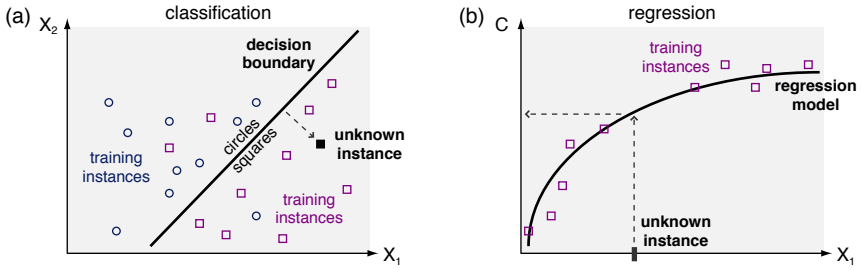

Figure 2.3: Illustration of supervised learning: (a) In classification, a decision boundary can be derived from training instances with known classes (open circles and squares) based on their features values, here for x_1 and x_2. The boundary decides the class of unknown instances. (b) In regression, a regression model can be derived from training instances (represented by the feature x_1) with known value for the target variable C. The model decides the values of all other instances.

In case of binary classification, machine learning algorithms seek for an optimal *decision boundary* that separates the instances of two classes, as illustrated in Figure 2.3(a). Multi-class classification is handled through approaches like one-versus-all classification (Hastie et al., 2009). The applications of classification in text mining are manifold. E.g., it denotes the standard approach to text classification (Sebastiani, 2002) and it is also often used to classify candidate relations between entities (Sarawagi, 2008). In all respective experiments below, we perform classification with a *support vector machine* (Witten and Frank, 2005). Support vector machines aim to maximize the margin between the decision boundary and the training instances of each class. They have been shown to often perform well (Meyer et al., 2003) while not being prone to adapt to noise (Witten and Frank, 2005).[10]

DECISION BOUNDARY

SUPPORT VECTOR MACHINE

In case of *regression*, the task is to assign a given data instance to the most likely value of a metric, continuous target variable (Witten and Frank, 2005). The result of learning is a *regression model* that can predict the target variable for arbitrary instances (cf. Figure 2.3(b)). We restrict our view to *linear regression* models, which we apply in Chapter 4 to predict the run-times of pipelines. In our experiments, we learn these models with stochastic gradient descent for efficiency purposes.

REGRESSION

REGRESSION MODEL

LINEAR REGRESSION

Unsupervised Learning In contrast to supervised learning, *unsupervised learning* is only given data instances without output information. As a consequence, it usually does not serve for predicting a target variable from an instance, but merely for identifying the organization and association of input data (Hastie et al., 2009). The most common technique in unsupervised

UNSUPERVISED LEARNING

[10] Some existing text analysis algorithms that we employ rely on other classification algorithms, though, such as *decision trees* or *artificial neural networks* (Witten and Frank, 2005).

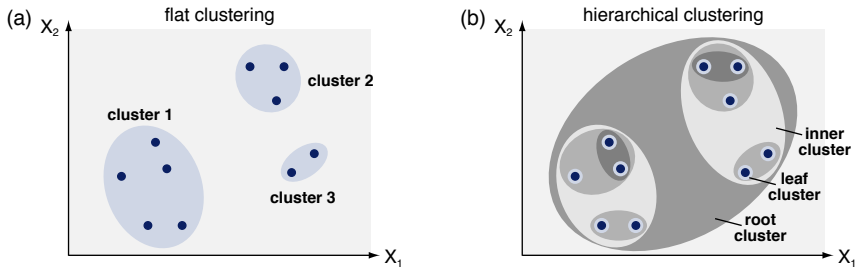

FIGURE 2.4: Illustration of unsupervised learning: (a) Flat clustering groups a set of instances into a (possibly predefined) number of clusters. (b) Hierarchical clustering creates a binary hierarchy tree structure over the instances.

CLUSTERING
CLUSTER

learning is *clustering*, which groups a set of instances into a possibly but not necessarily predefined number of *clusters* (Witten and Frank, 2005). Here, we consider only hard clusterings, where each instance belongs to a single cluster that represents some class. Different from classification, the meaning of a class is usually unknown in clustering, though. Clustering learns patterns in the similarities of instances based on similarity measures like those used in information retrieval (see above). The resulting model can assign arbitrary instances to one of the given clusters. In text mining, clustering is e.g. used to detect texts with similar properties.

Conceptually, two basic types of clustering exist, as shown in FIGURE 2.4.

FLAT CLUSTERING
HIERARCHICAL CLUSTERING

While *flat clustering* partitions instances without specifying associations between the created clusters, *hierarchical clustering* organizes instances in a hierachical tree (Manning et al., 2008). Each node in the tree represents a cluster of a certain size. The root cluster consists of all instances and each leaf refers to a single instance. A flat clustering can be derived from a hierarchical clustering through cuts in the tree. The tree is incrementally created by measuring distances between instances and clusters. To this end, a cluster is e.g. represented by its *centroid*, i.e., the average of all instances in the cluster (Manning et al., 2008). In general, both clustering types have certain advantages with respect to efficiency and cluster quality. We rely on hierarchical clustering in CHAPTER 5 for reasons discussed there. In particular, we perform *agglomerative hierarchical clustering* where the hierarchy is created bottom-up, beginning with the single instances (Manning et al., 2008).

CENTROID

AGGLOMERATIVE HIERARCHICAL CLUSTERING

Further Learning Types Some other machine learning types are used more or less frequently in text mining, part of which are variations of supervised learning. Sporadically, we talk about *semi-supervised learning* in this thesis, which targets at tasks where much input data is available, but little known training data. Intuitively, semi-supervised learning first derives patterns from the training data and then applies knowledge about these patterns to

SEMI-SUPERVISED LEARNING

find similar patterns in the other data (Chapelle et al., 2006). Some research in information extraction proposes *self-supervised learning* approaches that aim to fully overcome the need for known data by generating training data on their own (Banko et al., 2007). This can work when some output information is accessable without uncertainty. We present an according approach in Chapter 5. Also, we employ an entity recognition algorithm that relies on *sequence labeling* (cf. Appendix A). Sequence labeling classifies each of a sequence of instances, exploiting information about the other instances.

self-supervised learning

sequence labeling

While there are more learning types, like reinforcement learning, recommender systems or one-class classification, we do not apply them in this thesis and, so, omit to introduce them here for brevity.

Development and Evaluation

As discussed, text analysis aims to approximate unknown target functions, which map input texts to output information. To this end, both rule-based and statistical text analysis approaches are usually developed based on a collection of input texts with known properties. Still, the output information they produce will, in general, not always be correct. The reasons behind relate to the ambiguity of natural language (see above) and to the incompleteness and inexactness of the input data (Witten and Frank, 2005).

As a consequence, an empirical evaluation of the quality of a text analysis approach is of high importance and mostly closely connected to its development. Here, with quality we primarily refer to the effectiveness and efficiency of an approach, as outlined in Section 1.2.[11] While the concrete quality measures that are adequate for evaluation partly differ between the mentioned tasks from information retrieval, natural language processing, and data mining, in principle all three fields rely on similar methods (Manning et al., 2008; Jurafsky and Martin, 2009; Witten and Frank, 2005). In particular, experiments are performed, in which an approach is first developed on a collection of input texts. Then, its quality is measured on previously unseen input texts and compared to alternative approaches. In the following, we detail the outlined concepts for text analysis.

Text Corpora In this thesis, we approach text analysis in a *corpus linguistics* manner, i.e., we address all tasks based on the analysis of samples of real-world texts, called text corpora. A *text corpus* is a principled collection of texts that has been compiled to analyze a problem related to language (Biber et al., 1998). Here, we consider corpora that serve for the development of

corpus linguistics

text corpus

[11] Besides effectiveness and efficiency, we also investigate the *robustness* and *intelligibility* of text analysis in Chapter 5. Further details are given there.

text analyses (e.g. for sentiment analysis). Text corpora often contain annotations, especially annotations of the target variable that represents the output information to be inferred (e.g. the sentiment polarity of a text). In contrast to the annotations produced by text analysis algorithms, corpus annotations have usually been created manually in a cost-intensive *annotation process*. To avoid such a process, they can sometimes be derived from existing metadata (such as the author or star rating of a review). In both cases, they are seen as *ground truth* annotations (Manning et al., 2008).

ANNOTATION PROCESS

GROUND TRUTH

To allow for generalization, the compilation of texts in a text corpus usually aims to be *representative* for some target variable C, i.e., it includes the full range of variability of texts with respect to C (Biber et al., 1998). We discuss representativeness at the beginning of Chapter 5. For evaluation, also the distribution of texts over the values of C should be representative for the real distribution. For machine learning, though, a *balanced* distribution, where all values of the target variable are evenly represented, is favorable according to statistical learning theory (Batista et al., 2004).

REPRESENTATIVE

BALANCED

Effectiveness Text analysis approaches are mostly evaluated with respect to their effectiveness, which quantifies the extent to which output information is correct. Given a collection of input texts with ground truth annotations for the target variable C of a text analysis approach be given, the effectiveness of all approaches relevant in this thesis can be evaluated in the sense of a two-class classification task, i.e., whether the decision to produce each possible instance of C is correct or not.

We call the output instances of an approach the *positives* and all other possible instances the *negatives*. On this basis, four different sets can be distinguished (Witten and Frank, 2005): *True positives (TP)* are all positives that belong to the ground truth, *true negatives (TN)* are all negatives that do not belong to the ground truth, *false negatives (FN)* are all negatives that belong to the ground truth, and *false positives (FP)* are all positives that do not belong to the ground truth. Figure 2.5 illustrates the four sets.

POSITIVES

NEGATIVES

TRUE POSITIVES

TRUE NEGATIVES

FALSE NEGATIVES

FALSE POSITIVES

Given the four sets, effectiveness can directly be quantified with different measures whose adequateness depends on the given task. One measure is given by the *accuracy* a, which means the ratio of correct decisions:

ACCURACY

$$a = (|\mathrm{TP}| + |\mathrm{TN}|) \,/\, (|\mathrm{TP}| + |\mathrm{TN}| + |\mathrm{FP}| + |\mathrm{FN}|)$$

The accuracy is an adequate measure, when all decisions are of equal importance. This holds for many text classification tasks as well as for other text analysis tasks, in which every portion of an input text is annotated and, thus requires a decision, such as in tokenization. In contrast, especially in

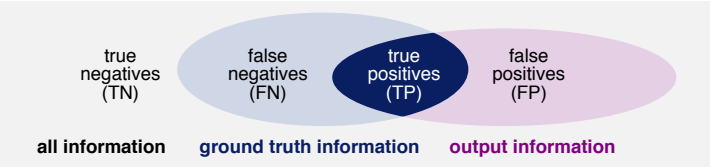

Figure 2.5: Venn diagram showing the four sets that can be derived from the ground truth information of some type in a collection of input texts and the output information of that type inferred from the input texts by a text analysis approach.

information extraction tasks like entity recognition, the output information usually covers only a small amount of the processed input texts. As a consequence, high accuracy can be achieved by simply producing no output information at all. Thus, accuracy is inadequate if the true negatives are of low importance. Instead, it seems more suitable to measure effectiveness in terms of the *precision* p and the *recall* r (Manning and Schütze, 1999):

PRECISION
RECALL

$$p = |\mathrm{TP}| \,/\, (|\mathrm{TP}| + |\mathrm{FP}|) \qquad \mathrm{r} = |\mathrm{TP}| \,/\, (|\mathrm{TP}| + |\mathrm{FN}|)$$

Precision quantifies the ratio of output information that is inferred correctly, while recall refers to the ratio of all correct information that is inferred. In many cases, however, achieving either high precision or high recall is as easy as useless. E.g., perfect recall can be obtained by producing all possible output information. If both high precision and high recall are desired, their harmonic mean can be computed, called the F_1*-score* (or F_1-measure), which rewards an equal balance between p and r (van Rijsbergen, 1979):

F_1-SCORE

$$f_1 = 2 \cdot \mathrm{p} \cdot \mathrm{r} \,/\, (\mathrm{p} + \mathrm{r})$$

The four defined effectiveness measures are used in a number of experiments in this thesis. In addition, we compute the mean *regression error* of numeric predictions in Chapter 5, which is defined as the average difference between a predicted and a correct value. Also, we mention the *labeled attachment score* once in Chapter 3, which denotes the proportion of fully correctly classified tokens in dependency parsing (Bohnet, 2010). Other effectiveness measures are not specified here for lack of relevance.

REGRESSION ERROR
LABELED ATTACHMENT SCORE

Efficiency Since we aim to perform text analysis on large amounts of input texts, not only effectiveness is important in this thesis, but also efficiency. In general, efficiency quantifies costs in terms of the consumption of time or memory (Cormen et al., 2009). While we sporadically discuss the effects of memory-related observations in the subsequent chapters, such as a high system load, we always refer to efficiency here as the *run-time* (also called running time) an approach takes to process a given input. We use the terms efficiency and *run-time efficiency* interchangeably from here on.

RUN-TIME
RUN-TIME EFFICIENCY

We quantify the efficiency of every algorithm A_i and pipeline Π in terms of two measures, both specified in seconds or milliseconds: First, the absolute *overall run-times* $t_i(D)$ and $t_\Pi(D)$ on an input text D, and second, the *average run-time per portion of text* (mostly, per sentence), $t(A_i)$ and $t(\Pi)$. All run-times are averaged over a defined number of runs (either 5 or 10) and complemented by their *standard deviation* σ. In some cases, we compute specific run-times (say, the training time), as defined where given.

OVERALL RUN-TIME
AVERAGE RUN-TIME PER PORTION OF TEXT
STANDARD DEVIATION

Experiments In corpus linguistics, the general method to develop and evaluate both rule-based and statistical text analysis approaches is to perform experiments based on a split of a text corpus into different *datasets*. One (or the union of several) of these datasets is analyzed manually or automatically for the development, and the others are processed to evaluate a developed approach (Jurafsky and Martin, 2009).[12] We realize the process underlying this method in the following two ways in the thesis at hand, both of which are very common in statistical evaluation (Witten and Frank, 2005).

DATASET

In most cases, we split a given text corpus into a training set, a validation set, and a test set, as illustrated in FIGURE 2.6(a).[13] After developing an approach on the *training set*, the quality of different configurations of the approach (e.g. with different feature vectors or learning parameters) is iteratively evaluated on the *validation set*. The validation set thereby serves for optimizing the approach, while the approach adapts to the validation set. The best configuration is then evaluated on the *test set* (also referred to as the held-out set). A test set represents the unseen data. It serves for estimating the quality of an approach in practical applications.[14]

TRAINING SET
VALIDATION SET
TEST SET

The described method appears reasonable when each dataset is of sufficient size and when the given split prevents from bias that may compromise the representativeness of the respective corpus. In other cases, an alternative is to perform (stratified) *n-fold cross-validation* (Witten and Frank, 2005). In n-fold cross-validation, a text corpus is split into n (e.g. 10) even folds, assuring that the distribution of the target variable is similar in all folds. The development and evaluation then consist of n runs, over which the measured quality of an approach is averaged. In each run i, the i-th fold is used for evaluation and all others for development. Such a split is shown in FIGURE 2.6(b). We conduct according experiments once in CHAPTER 5.

N-FOLD CROSS-VALIDATION

[12] The development of statistical approaches benefits from a balanced dataset (see above). This can be achieved through either undersampling minority classes or oversampling majority classes. Where needed, we mostly perform the latter using random duplicates.

[13] Many text corpora already provide an according corpus split, including most of those that we use in our experiments (cf. APPENDIX C).

[14] In some of our efficiency experiments, no parameter optimization takes place. We leave out the use of validation set in these cases, as pointed out where relevant.

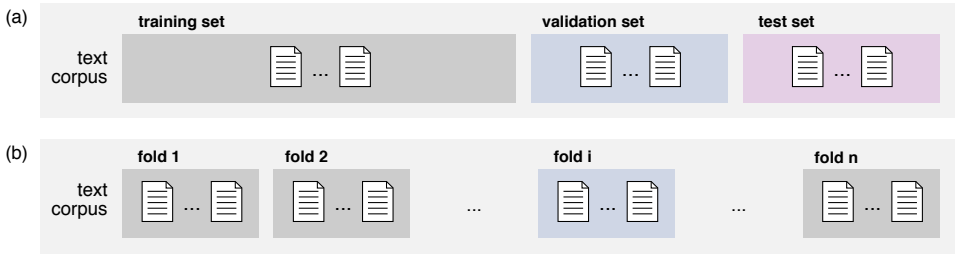

Figure 2.6: Illustration of two ways of splitting a text corpus for development and evaluation: (a) A training set is used for development, a validation set for optimizing parameters, and a test set for the final evaluation. (b) Each fold i out of n folds serves for evaluation in the i-th of n runs, while all others are used for development.

Comparison The measured effectiveness and efficiency results of a text analysis approach are usually compared to alternative ways of addressing the given task in order to assess whether the results are good bad. For many tasks, an upper-bound ceiling of effectiveness is assumed to be the effectiveness a human would achieve (Jurafsky and Martin, 2009).[15] For simplicity, effectiveness is thus often measured with respect to the human-annotated ground truth. While there is no general upper-bound efficiency ceiling, we see in the subsequent chapters that optimal efficiency can often be determined in a given experiment setting. We call every upper-bound ceiling of a quality measure the *gold standard* and define it where needed. GOLD STANDARD

For interpretation, results are also checked whether they are significantly better than some lower bound *baseline* (Jurafsky and Martin, 2009). E.g., an accuracy of 40% in a 5-class classification task may appear low, but it is still twice as good as the accuracy of guessing. The standard way to determine lower bounds is to compare an evaluated approach with one or more approaches that are trivial (like guessing), standard (like a bag-of-words approach in text classification), or known from the literature. We compare our approaches to according baselines in all our experiments in Chapters 3 to 5. In these experiments, we mostly consider complex text analysis processes realized by pipelines of text analysis algorithms, as presented next. BASELINE

2.2 Text Analysis Tasks, Processes, and Pipelines

In Section 1.2, we have roughly outlined that text mining requires task-specific text analysis processes with several classification, extraction, and similar steps. These processes are realized by text analysis pipelines that infer output information from input texts in order to satisfy a given infor-

[15] Some exceptions to the truth of this assumption exist, of course. For instance, authorship attribution (see above) is expected to be often hard for humans.

mation need. Since text analysis pipelines are in the focus of all approaches proposed in this thesis, we now explain the outlined concepts of text analysis more comprehensively and we illustrate them at the end. Thereby, we define the starting point for all discussions in Chapters 3 to 5.

Text Analysis Tasks

As specified in Section 1.2, we consider text analysis tasks in which we are given input texts and an information need to be addressed. The goal is to infer certain output information from the input texts that is relevant with respect to the information need. Here, we detail basic concepts behind such tasks. An extension of these concepts by quality criteria to be met follows in Chapter 3 after discussing the optimality of text analysis pipelines.

COLLECTION OF TEXTS
STREAM OF TEXTS

Input Texts In principle, the input we deal with in this thesis may be either given in the form of a *collection of texts* or a *stream of texts*. The former denotes a set of natural language texts $\{D_1, \ldots, D_n\}$, $n \geq 1$, usually compiled with a purpose, like a text corpus (see above). With the latter, we refer to continuously incoming natural language text data. We assume here that such data can be split into logical segments $D_1, D_2, \ldots$ (technically, this is always possible anyway). Given that the speed of processing a stream can be chosen freely, we can therefore deal with collections and streams in the same way except for the constraint that streaming data must be processed in the order in which it arrives. We denote both a collection and a stream as $\mathbf{D}$.

UNSTRUCTURED TEXT

We see a single text $D \in \mathbf{D}$ as the atomic input unit in text analysis tasks. While no general assumptions are made about the length, style, language, or other properties of D, we largely restrict our view to fully *unstructured texts*, i.e., plain texts that have no explicit structure aside from line breaks and comparable character-level formattings. Although text mining may receive several types of documents as input, such as HTML files in case of web applications, our restriction is not a limitation but rather a focus: Most text analysis approaches work on plain text. If necessary, some content extraction is, thus, usually performed in the beginning that converts the documents into plain text (Gottron, 2008). Besides, some of our approaches in the subsequent chapters allow the input texts to already have annotations of a certain set of zero or more information types $\mathbf{C}_0$, which holds for many text corpora in computational linguistics research (cf. Section 2.1).

Output Information In Section 2.1, different information types have been mentioned, e.g. tokens, part-of-speech tags, concrete types of entities and relations, certain text classification schemes, etc. In general, an information type $C = \{c_1, c_2, \ldots\}$ denotes the set of all pieces of information $c \in C$ that