1
Machine learning for categorisation of speech utterances

Amparo Albalate, David Suendermann, Roberto Pieraccini and Wolfgang Minker

1.1
Abstract

In this chapter we discuss several methods for the categorisation of call utterances in the framescope of automated troubleshooting agents. Automated agents are spoken language dialog systems of the 3rd generation, oriented to perform technical support tasks over the phone in a similar way as human agents do. One of the issues to complete to this aim is the identification of the problem experienced by the caller out of caller's utterance, which is currently addressed by statistical classificaton methods. In this chapter, two different approaches to the categorisation of transcribed utterances are described. First, statistical categorisers which require minimal supervision degrees, in terms of labeled samples, are proposed. A nearest neigbour algorithm feeded with only one labeled utterance per problem category is applied in combination with appropriate feature extraction schemes – semantic term clustering. Secondly, different classifiers typically applied to the categorisation of text documents are compared, also for different sizes of the labeled sets. In this sense, a vector model is compared to a probabilistic approach – the Naïve Bayes classifier. All described techniques are evaluated and tested with two different corpora gathered from real interactions of commercial troubleshooting agents with callers. In general terms, the main objective of the chapter is to provide the reader an overview to the fields of pattern recognition and text classification, whilst focusing on the practical categorisation of utterances for a modern dialog system application – the problem solving domain.

1.2
Introduction

As a result of an accelerated technological development and especially due to the progressive advances in the field of automated speech recognition, first

Titel. Author(s)

ISBN: 3-527-XXXXX-X

Spoken Language Dialog Systems (SLDSs) emerged in the mid 1990s as a new, important form of human-machine communication.

As their name suggests, SLDSs are interactive, voice-based interfaces between humans and computers, which allow humans to carry out tasks of diverse complexity (travel ticket reservations, bank transactions, information search or problem solving, etc.).

The typical architecture of an SLDS [28] is depicted in Fig. 1.1. Input acoustic vectors generated from the speech signal are first processed by an Automatic Speech Recogniser (ASR), resulting in a raw text transcription[1] of the input utterance. Subsequently, the transcribed text is interpreted in a semantic analysis block which extracts the utterance meaning in form of an appropriate semantic structure. This semantic representation is processed by the dialog manager which also communicates directly with an external application, namely a database interface. The dialog manager keeps control of the overall interaction progress towards the task completion. During this process, the user may be queried for confirmations, disambiguations, necessary additional information, etc. Finally, the interaction result is presented to the user in form of speech (text-to-speech synthesis or pre-recorded prompts), text, tables or graphics.

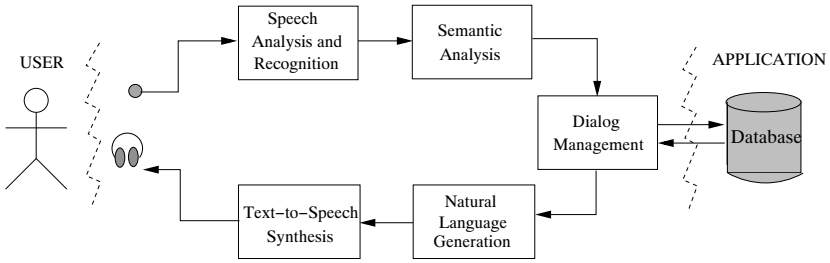

Fig. 1.1 Overview of an SLDS.

Among the SLDS modules, speech recognition and semantic analysis play a decisive role for the global system performance [6]. In particular, this chapter deals with the semantic analysis block, often referred to as *natural language understanding*. The extracted *semantics* from each user utterance can be viewed as an internal knowledge representation used (by the dialog manager) to trigger a certain action in the context of a particular task [31].

In first and second generation SLDS , frequently used in applications such as banking and travel reservations, semantic analysis commonly relies on the definition of semantic or case-frame *grammars* [4]. A semantic gramar

1) Most probable sequence of words detected

formalism provides a model for the sentence structure in terms of semantic constituents: words or phrases. The semantic analysis *decodes* the text of an input utterance by extracting the correspondences between the sentence constituents and their semantic labels. For example, in the framework of a flight booking application, the user utterance "*I would like to fly from Munich to New York on July,* 24^{th}" may be decoded into the following semantic sequence: <book> (airport-origin)(airport-destination)(depart-day)(depart-month).

For the grammar implementation, two major tendencies exist: in a rule-based approach, a set of grammar rules is manually defined for a specific task or application. Rule-based methods provide best performance for a restricted task for which they are originally designed. However, these methods turn out to be inflexible regarding their adaptation and portability to new application domains. Alternatively, in *data-oriented* approaches, stochastic models are used, such as Hidden Markov Models [38], which automatically infer the model parameters from training corpora of semantic representations. These techniques are more flexible and portable to different domains. Examples of systems using rule-based and stocastically-based parsing principles are the ATR translation system from Japanese to English (SL-TRANS) [33] and the AT&T-CHRONUS (Conceptual Hidden Representation of Natural Unconstrained Speech) speech understanding system [27], respectively.

However, third generation SLDSs , deployed in applications dealing with problem solving, education and entertainment, have shown higher levels of complexity. In this chapter, we focus on the problem solving domain, in particular on automated troubleshooting agents. These agents are specially designed to perform customer care issues over the telephone in a similar way as human agents do.

Today, natural language understanding is typically performed by a speech recognition module followed by a speech utterance classifier . Such classifiers are a more sophisticated replacement of menu-based systems using dual-tone multi-frequency (DTMF) [18] technology (*... push 1 for billing, push 2 for sales ...*) or speech-recognition-based directed dialog (*... you can say billing, sales, or ...*). These simple solutions are often not practical for several reasons:

- In certain applications, the number of classes can be too large to be handled in a single menu. Even succession of menus hierarchically structured would prove unwieldy with hundreds of classes, not to mention the bad caller experience when five or six menu levels are required to reach the correct routing point.

- Even when prompted with a clear menu, callers often describe the reason why they are calling in their own words, and that may not be covered by the rule-based grammar typically used with directed dialog systems.

- For complex domains, callers may not understand or be familiar with the terms used in the menu. For example in response to the prompt: *Do you have a hardware, software, or configuration problem?*, they may respond unexpectedly (*My CD-ROM does not work!*) or choose one of the options at random without really knowing if it applies to their case.

For these reasons, state-of-the-art troubleshooting agents leave the dialog initiative to the users by presenting an open welcome message: *"please briefly describe the reason for your call"* [1]. Unconstrained, natural language user responses describing the general problem or symptom they experience are then classified by an speech utterance classifier mapping the user utterance into one of a set of predefined categories [14].

Supervised statistical classifiers are algorithms trained with a corpus of transcribed utterances and their associated problem categories. The parameters learned in training phase are applied to predict the classes of new utterances, not necessarily observed in the training corpus. A crucial factor on which a classifier's effectiveness depends is the size of available data for training.

However, the significant cost of hand-labelling a large amount of training data is one of the main problems associated with the use of such classifiers. Achieving appropriate classification performance even with small training sets recently became focus of research in the field [5]. Besides, the set of categories used for data-labelling is subject to alteration: It is not rare to observe situations in which the set of problems handled by the automated agents needs to be updated. In such cases, algorithms which require only few training data can be helpful to rapidly adapt the system.

In this chapter, we first provide an overview on the utterance categorisation model and propose different schemes which use only one labelled example per category. With these minimal training data, considerable degradation of the categorisation performance is expected with respect to categorisers that make use of large labelled corpora. One main reason is that semantic variability may not be adequately captured in small labelled sets. We therefore analyse word clustering as a mean to extract semantic relationships of words and in consequence boost the classification effectiveness. A similar task in the field of information retrieval is the efficient search of information in the Internet. In fact, one of the first applications of word clustering is the lexical term expansion of user queries to search engines with automatically discovered synonyms of the original query terms [42].

We also provide a comparison of formulations used in text processing application for estimating the different relevance of terms. Term scoring was applied to the categorisation of utterances with different numbers of labelled examples.

The chapter organisation is as follows: an overview to general pattern recognition and its application to the categorisation of texts is given in Sections 1.3 and 1.4. In Section 1.5 a description of the utterance corpora used in our experiments is provided. The utterance preprocessing is explained in Section 1.6. Details about feature extraction and term weighting are outlined in Sections 1.7 and 1.8 respectively. Finally, we evaluate the described algorithms in Section 1.9 and draw conclusions in Section 1.10.

1.3 An overview on pattern recognition

Pattern recognition is an important problem addressed by scientists in a number of research fields: biology, geography, engineering, computer science, artificial intelligence, etc. [20]. In pattern recognition, patterns are defined as entities which can be subjected to classification. This is possible as long as their similarity can be calculated. Examples of patterns are genes, human faces, handwritten characters, or texts.

The classification task consists of (i) the mapping of patterns into one or more classes out of a pre-defined category set (supervised classification or discriminant analysis), or (ii) the grouping of patterns into previously unknown classes according to their affinities (unsupervised classification or clustering). In the latter case, the classes are also detected as a result of the classification process. A typical pattern recognition scheme is shown in Figure 1.2.

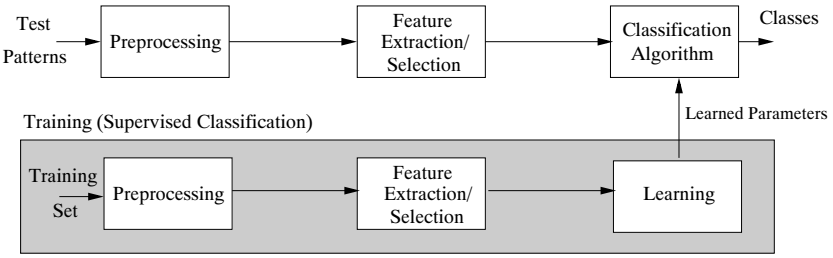

Fig. 1.2 Pattern recognition scheme. For supervised classification, a training module is required.

In supervised classifiers, pattern recognition operates in two separated modes: *training* and *classification* or *test*. In the case of unsupervised classification, the learning step is absent. In addition to the modes, one distinguishes between three phases: Preprocessing, feature selection/extraction and classification.

Preprocessing, also known as preparation, aims at optimising the representation and quality of the input observations in order to produce reliable data for statistical analysis [37]. This process involves operations such as segmentation, normalisation and elimination of noise or irrelevant information.

A segmentation stage decomposes the input data into pieces, thereby enabling the multidimensional representation of patterns. In certain cases, input objects are already presented segmented as a set of measurements captured by an array of sensors (for example, temperature and humidity in classification of metereological phenomena). However, in many other situations, the objects to classify are the result of individual acquisitions. This is the case of images in computer vision. A data segmentation may be used here to split digital images into M pixels or blocks, so that an image can be observed, for example, as an M dimensional array of pixel intensities. The output dimensions obtained after segmentation are also termed *classification features*, since they represent different properties of the objects to classify. In consequence, patterns are also referred to as *feature vectors*.

Normalisation procedures can be applied to features or patterns. Feature normalisation is specially convenient and necessary if the classification features represent different object attributes represented in different scales. Common feature normalisation techniques are linear scaling to unit variance, transformation to uniform [0-1] random variables and rank normalisation, among other methods [3]. Moreover, pattern normalisation applies to the feature values inside an individual pattern. An example is the normalisation of image intensities for object recognition in images.

Feature selection and extraction techniques help in reducing the dimensionality of feature sets. As it is broadly accepted, an optimal feature set should capture the relevant characteristics of the data in the most compact way.

Feature *selection* aims at retaining the subset of the original features that best represents the input patterns. Typically, this process is carried out by sorting the initial features according to their relevance and filtering out those features which do not exceed a minimum relevance threshold. The resulting feature vectors are thus the projections of the original patterns over the selected feature sub-space. In contrast, feature *extraction* performs a transformation of the input pattern vectors into a different feature space through certain statistical analysis of the input data. Examples of feature extraction techniques are principal components analysis (PCA), independent component analysis (ICA) or feature clustering [26]. The feature selection/extraction module has proven highly important for pattern recognition. A correct scheme may not only help reducing computational costs associated with very high dimensional data sets, but also increase the classification effectiveness.

Finally, the **classification algorithm** maps input feature vectors to output classes. Supervised algorithms rely on the existence of training sets with labelled examples. The mapping is typically defined by a certain number of parameters whose values are usually adjusted to a training data set during the learning phase. Some examples of supervised classifiers include, among others, the Naïve Bayes classifier, polynomial classifiers, neural networks or support vector machines.

Unsupervised techniques are suitable when no labelled examples are available. The output classes/groups are not known a-priori, but detected during the classification process. Hierarchical and partitioning clustering algorithms are used to group the input patterns according to distance-based criteria. Hierarchical approaches build the cluster solution gradually, resulting in cluster hierarchy structures or *dendograms*. Two kind of hierarchical algorithms can be distinguished, depending on the dendogram construction methods: aglomerative (bottom up) and divisive (top down) [19,21,24]. In contrast, partitioning approaches learn a flat cluster structure, typically through an iterative search for the optimum of the criterion function (K-means, K-medoids, etc.) [17,34]. More recent approaches have been developed to discover dense regions in a dataspace. Usually, the density notion is represented by two parameters, *minpts* and ε, denoting the minimum number of points to be enclosed in a ε-radius neigbourhood of certain objects called core points (DBscan [13], Optics [29], Denclue [15], Clique [2], etc.). These algorithms are resistant to outliers[2] and more flexible than distance-based approaches, insofar clusters of irregular shapes and sizes can be detected [43]. Further, if the set of observations can be drawn from an underlying probabilistic distribution, model-based approaches can be applied in order to fit a probabilistic model to the input patterns. A common example is the Expectation Maximisation algorithm, used to fit a mixture of gaussians to a dataset [11].

A compromise between supervised and unsupervised techniques are semi-supervised approaches [9]. These methods make use of both labelled and unlabelled data for training. In *co-training* algorithms, two or more supervised classifiers are applied to different subsets of the original feature set [5]. A new training data set is generated following a confidence evaluation of the classification results (e.g. agreement between classifiers). The main condition for the use of co-training approaches is the statistical independence between feature subsets used by the classifiers. Another kind of semi-supervised learning are clustering algorithms in which certain constraints about the input data are manually defined (*Clustering with constraints*) [41]. The constraints spec-

2) Outliers are *noise* patterns which do not belong to any cluster, but fall in the regions between two or more clusters. Outliers are often unreliable patterns which need to be discovered and accordingly treated.

ify wether two data instances must or cannot be linked together in a single cluster.

1.4 Utterance classification as a text classification problem

Since speech utterances are transcribed into text by ASR, utterance-to-symptom categorisation is a particular case of text classification, traditionally applied to documents. In this section, we describe how pattern recognition is applied to text and, in particular, to utterance classification.

During preprocessing, all words in a text corpus are reduced to units of semantic meaning: *stems* or *lemmas*. As a next step, an n-gram model[3] can be applied to extract and count subsequences of terms up to length n. A particular case is the uni-gram model where only single words are extracted, ignoring any possible order in which the words appear in the text. Due to their simplicity and adequate performance for classification, uni-grams are possibly the most frequently used approach for the representation of texts. When used for the representation of texts or utterances, uni-gram structures are commonly referred to as *bags of words*. Usually, texts are represented as vectors over a basis of terms or n-grams in what is called *vector space model* [40]. A simplistic approach is to use binary vector components denoting the presence or absence of the respective terms in a text. Also, other vector components may be used to reflect term frequency counts in the text, or terms' discriminative significance estimated through relevance scores. A popular metric for estimating a word's relevance is the *term frequency - inverse document frequency (TF-IDF)* (Refer to section 1.8.2 for more details about term scores).

Common feature selection algorithms are based on the aforementioned relevance scores in order to filter out unimportant terms that do not exceed a relevance treshold. In contrast, feature extraction approaches provide a transformation of the initial term features into a new feature space in which semantic effects related to terms can be mitigated, namely synonymy and polysemy. Synonymy refers to the fact that multiple terms can be used to denote a single concept - words with identical meaning[4]. Polysemy, on the other side, indicates the existence of terms with multiple related meanings, which can there-

3) n-gram model is a sentence structure specification based on the assumption that the probability of occurrence of a given word is conditioned upon the prior $N-1$ terms. While the n-gram specification is of high relevance for the development of grammars and lexical parsers, it is less important for capturing the underlying semantics (meaning) of texts.

4) In text processing applications, the synonymy concept is used in a general sense, to indicate not only terms with identical meaning but also terms with *similar* meaning (soft synonyms)

Tab. 1.1 Corpus definition. Number of categories (L) and number of utterances of test and training sets

Corpus	Number of Symtoms	Training (# utt.)	Test (# utt.)
Internet	28	3313	31535
Cable TV	79	10000	10000

fore be observed in different contexts. These semantic artifacts are pointed out as one of the fundamental problems to be faced in text categorisation, as they introduce a clear obstacle in capturing the semantic proximity between texts [10]. Attempts to address synonymy and/or polysemy have relied on Latent Semantic Analyis (LSA) [10, 16] and feature clustering [26], among other methods.

1.5 Utterance corpus description

For the experiments and results reported in the following sections, we used two corpora of transcribed and annotated caller utterances gathered from user interactions of commercial troubleshooting agents of the Internet and Cable TV domains. Some examples of transcribed utterances are:

- Internet troubleshooting:
  - *Internet was supposed to be scheduled at my home today.*
  - *I'm having Internet problems.*
- Cable TV troubleshooting:
  - *I have a bad picture quality.*
  - *I don't get HBO channel.* (ChannelMissing)

Further details about the corpora including the number of categories considered in this work as well as the dimensions of training[5] and test sets are shown in Table 1.1.

5) Note that, since the approaches described in this chapter make reference to small amounts of examples, we refer to the part of the available corpora used to select the categorisers examples as training set and, if necessary, perform certain statistical analyses which do not require the use of utterance labels.

1.6 Utterance preprocessing

The preprocessing module consists of part-of-speech (POS) tagging, morphological analysis, stop word filtering, and bag-of-words representation.

First, the Stanford POS tagger [23] performs an analysis of each sentence and tags the words with their lexical categories (POS tags).

Subsequently, a morphological analyser [30] is applied to reduce the surface word forms in utterances into their corresponding lemmas.

As a next step, stop words are eliminated from the lemmas, as they are judged irrelevant for the categorisation. Examples are the lemmas *a, the, be, for*. In this work, we used the SMART stop word list [7] with small modifications: in particular, we deleted confirmation terms (*yes* and *no*) from the list, whereas words typical for spontaneous speech (*eh, ehm, uh*) were treated as stop words.

For example, the input utterance *My remote control is not turning the television* is transformed through the described steps (POS tagging, morphological analysing and stop word filtering) as follows:

- POS tagging: my/PRP remote/JJ control/NN is/VBZ not/RB turning/VBG the/DT television/NN[6]
- Morphological analysis: My remote control be not turn the television
- Stop word filtering: remote control not turn television

The salient vocabulary is then defined as the set of distinct lemmas in the preprocessed training utterances: $W = (w_1, \ldots, w_D)$. The vocabulary dimensions in Intenet and Cable TV troubleshooting corpora are $D = 1614$ and $D = 1022$, respectively.

Finally, the lemmas for each utterance are combined as a bag of words. I.e., each utterance is represented by an (unweighted) D-dimensional vector whose binary elements represent the presence/absence of the respective vocabulary element in the current utterance:

$$BW = (b_1, \ldots, b_D). \tag{1.1}$$

1.7 Feature extraction based on term clustering

One of the simplest categorisation algorithms is the nearest neighbour (NN) approach. Given a set of M labelled examples per category (prototypes), the NN algorithm assigns each input pattern to the category of the closest prototype. In this work, we only use one prototype per category ($M = 1$), selected

6) For a detailed inventory of POS tags used by the Stanford parser and their meanings, please refer to the parser homepage: *http://nlp.stanford.edu/software/lex-parser.shtml*

from the training corpus. One should therefore expect a degradation of the classifier performance with respect to categorisers making use of all utterance labels in the training set. This is partly due to the prevalence of synonymy and polysemy, which may be insufficiently represented in a small amount of prototypes. Also what is considered to belong to a class can be arbitrary and is up to the system design and what the classification result is used for further down in the application[7].

In effect, by using one labelled utterance per category, the effective vocabulary available to the categoriser is reduced to less than 10% of the vocabulary in the training set (W). This results in a large amount of utterances mapped to a *nomatch* class, provided the existence of out-of-vocabulary terms. As example, we want to look at the category representing a problem related to sound (NoSound). One would select a typical caller utterance reporting this problem, *no sound*, as the category prototype. However, the user may utter other alternatives, such as *"problem with volume"* or *"lost audio"*, which cannot be matched to the desired prototype due to the bag-of-words' orthogonalities (absence of overlapping terms with the prototype). This problem could be partially solved if one could detect that *sound* has a similar meaning to *audio* or *volume*.

The feature extraction methods described in the following paragraphs aim at capturing semantic relationships between words. We analyse two approaches to the classification of words based on hard and fuzzy clustering.

In hard clustering, each input pattern is unequivocally allocated to one output cluster. This approach may be adequate for capturing semantically related terms (e.g. synonyms) in output *semantic* classes. In contrast, a soft clustering algorithm associates the input patterns to all output classes through a matrix with membership degrees. If a considerable amount of polysemous terms (with several related meanings) is present in the input data, fuzzy techniques should then be more appropriate. An overview on utterance categorisation based on term clustering is shown in Figure 1.3.

After the feature extraction phase, each input bag of words (BW) is accordingly transformed into a feature vector F. Details of feature extraction based on hard and fuzzy clustering are discussed in the following sub-sections.

7) E.g., the corpora used in this study contain a class for multiple symptoms (like *my picture is out, and I have no sound*) which is purposely omitted when the classifier is trained to catch such an utterances with one of the single-symptom classes (such as *NoPicture* and *NoSound* in the above example). It is extremely unlikely that such a class would be automatically isuolated as it potentially contains contributions from all the other classes.

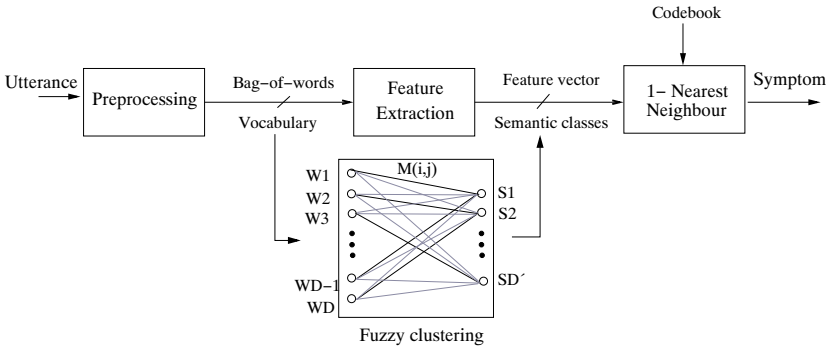

Fig. 1.3 Utterance categorisation components. For feature extraction, hard and fuzzy approaches to term clustering are compared. Hard clustering provide a hard mapping of each vocabulary term pattern into a single output semantic class (black traces). In contrast, a fuzzy clustering provides a *fuzzy* or *soft* association of each pattern to the output classes through a membership matrix (grey lines). Hard clustering can also be observed as a particular case of fuzzy clustering, where pattern memberships are either 1 or 0.

1.7.1 Term vector of lexical co-occurrences

A frequently reported problem to word clustering is the adequate representation of word lemmas in vector structures so that mathematical (dis)similarity metrics applied to term vectors can reflect the terms' semantic relationships.

In this respect, there are, among other, two criteria in the literature which attempt to explain the main characteristics of semantically related terms.

1. **First order co-occurrence**: two words are similar to the degree that they co-occur or co-absent in texts [25, 42].

2. **Second order co-occurrence**: Two words are similar to the degree that they co-occur with similar words [36].

The first order co-occurrence criterion is adequate for text documents where a semantic variability can be observed inside a document. In contrast, semantically related terms rarely occur inside a sentence. Thus, a second order co-occurrence criterion seems to be more appropriate for detecting terms' semantic proximities from an utterance corpus.

Consequently, each vocabulary term w_i is represented in a D-dimensional vector of lexical co-occurrences:

$$W_i = (c_{i1}, \ldots, c_{iD}) \tag{1.2}$$

wherein the constituents c_{ij} denote the co-occurrence of the terms w_i and w_j, normalised with respect to the total sum of lexical co-occurrences for the term w_i:

$$c_{ij} = \frac{nc_{ij}}{\sum_{k \neq i} nc_{ik}}. \tag{1.3}$$

Here, nc_{ij} denotes the total number of times that w_i and w_j co-occur. Finally, in order to extract the terms' semantic dissimilarities, we apply the Euclidean distance between term vectors.

1.7.2 Hard term clustering

A hard clustering algorithm places each input pattern into a single output cluster. Based on the complete-link criterion [21], the proposed term clustering produces a partition of the vocabulary terms given an input user parameter, the maximum intra-cluster distance d_{th}:

1. Construct a dissimilarity matrix U between all pairs of patterns. Initially, each pattern composes its individual cluster $c_k = \{w_k\}$.
2. Find the patterns w_i and w_j with minimum distance in the dissimilarity matrix.
   - If the patterns found belong to different clusters, $c_a \neq c_b$, and $(U_{max}(c_a, c_b)) \leq d_{th}$, where $d_{max}(c_a, c_b))$ is the distance of the furthest elements in c_a and c_j, merge clusters c_a and c_b.
   - Update U so that $U_{ij} = \infty$.
3. Repeat step 2) while $U_{min} \leq d_{th}$ or until all patterns remain assigned to a single cluster.

As a result of the hard term clustering algorithm, different partitions of the vocabulary terms are obtained, depending on the input parameter d_{th}. Because the elements in each cluster should indicate terms with a certain semantic affinity, we also denote the obtained clusters as *semantic classes*. Table 1.2 shows examples of clusters produced by this algorithm.

After hard term clustering, the bag of words remains represented in a binary feature vector F_{hard}:

$$F_{hard} = (b_{f_1}, b_{f_2}, \ldots, b_{f_{D'}}) \tag{1.4}$$

where the b_{f_i} component denotes the existence of at least one member of the i^{th} extracted class in the original bag of words.

Tab. 1.2 Example utterances of semantic classes obtained by hard term clustering for $d_{th1} = d$ on a text corpus comprising 30,000 running words from the cable televison troubleshooting domain; average number of terms per cluster is 4.71; number of extracted features is 1458

speak, talk
operator, human, tech, technical, customer, representative, agent, somebody, someone, person, support, service
firewall, antivirus, protection, virus, security, suite, program, software, cd, driver
reschedule, confirm, cancel, schedule
remember, forget
webpage, site, website, page, web, message, error, server
megabyte, meg
technician, appointment
update, load, download
boot, shut, turn
user, name, login, usb
area, room, day

Disambiguation If applied to bags of words or feature vectors extracted from hard term clusters, the NN classifier rejects a considerable number of ambiguous utterances for which several candidate prototypes are found[8]. A disambiguation module was therefore devised to resolve the mentioned ambiguities and map an ambiguous utterance to one of the output categories.

First, utterance vectors with more than one candidate prototype are extracted. For each pattern, we have a list of pointers to all candidate prototypes. Then, the terms in each pattern that cause the ambiguity are identified and stored in a competing term list.

As an example, let us consider the utterance *I want to get the virus off my computer* which, after pre-processing and hard term clustering, results in the feature set *computer get off virus*. Its feature vector has maximum similarity to the prototypes *computer freeze* (category *CrashFrozenComputer*) and *install protection virus* (category *Security*). The competing terms that produce the ambiguity are in this case the words *computer* and *virus*. Therefore, the disambiguation among prototypes (or categories) is here equivalent to a disambiguation among competing terms. For that reason, as a further means of disambiguation, we estimate the *informativeness* of a term w_i as shown in Equation 1.5:

$$I(w_i) = -(log(Pr(w_i)) + \alpha \cdot log(\sum_{\substack{j \\ L_j=N}} c_{ij} Pr(w_j))) \tag{1.5}$$

8) Candidate prototypes are such prototypes which share maximum proximity to the input utterance. This happens specially when the similarity metric between the vectors results in integer values, e.g. in the case of using the inner product of binary vectors as the aforeintroduced bags of words and feature vectors are.

where $Pr(w_i)$ denotes the maximum-likelihood estimation for the probability of the term w_i in the training corpus, and L_j refers to the part-of-speech (POS) tag of w_j.

As it can be inferred from Equation 1.5, two main factors are taken into account in order to estimate the relevance of a word for the disambiguation:

a) the word probability and

b) the terms' co-occurrence with frequent nouns in the corpus.

The underlying assumption that justifies this second factor is that words representative of problem categories are mostly nouns and appear in the corpus with moderate frequencies. The parameter α is to control the trade-off between the two factors. Reasonable values are in the range of $(\alpha \in [1, 2])$ placing emphasis on the co-occurance term; for our corpus, we use $\alpha = 1.6$ we found best-performing in the current scenario.

Finally, the term with highest informativeness is selected among the competitors, and the ambiguous utterance vector is matched to the corresponding prototype or category.

1.7.3 Fuzzy term clustering

The objective of the fuzzy word clustering used for feature extraction is a fuzzy mapping of words into semantic classes and leads to the membership matrix M representing this association.

1.7.4 Pole-based overlapping clustering

In the PoBOC algorithm [8], two kinds of patterns are differentiated: poles and residuals.

Poles are homogeneous clusters which are as far as possible from each other. In contrast, residuals are outlier patterns that fall into regions between two or more poles. The elements in the poles represent monosemous terms, whereas the residual patterns can be seen as terms with multiple related meanings (polysemous).

The PoBOC algorithm is performed in two phases: (i) pole construction, and (ii) multiaffectation of outliers.

In the **pole construction** stage, the set of poles $\{P\} = \{P_1, \cdots, P_{D'}\}$ and outliers $\{R\}$ are identified and separated. Poles arise from certain terms with maximal separation inside a dissimilarity graph which are therefore known as the pole generators.

In the **multi-affectation** stage, the outliers' memberships to each pole in $\{P\}$ are computed. Finally, the term w_i is assigned a membership vector to each P_j

pole as follows:

$$M_{ij} = \begin{cases} 1, & \text{if } w_i \in P_j \\ 1 - d_{av}(W_i, P_j)/d_{max} & \text{if } w_i \in \{R\} \\ 0, & \text{otherwise} \end{cases} \tag{1.6}$$

where $d_{av}(W_i, P_j)$ denotes the average distance of the w_i word to all objects in P_j, and d_{max} refers to the maximum of the term dissimilarity matrix.

For computing the semantic dissimilarity of terms, experiments with both Euclidean and cosine distances[9] were carried out.

PoBOC with fuzzy *C*-medoids The fuzzy *C*-medoids algorithm (FCMdd) [22] computes the fuzzy membership matrix M starting from an initial choice of cluster representatives or *medoids*. We initialise the algorithm with the D' pole generators ($C = D'$) obtained at the pole construction phase of the PoBOC scheme. The final solution for the membership matrix M is then reached through the iterative repetition of two steps: (i) (re)calculation of pattern memberships to the D' classes, and (ii) recomputation of the cluster medoids. The membership update of the term W_i to the j^{th} class is defined as:

$$M_{ij} = \frac{\left(\frac{1}{d(W_i,C_j)}\right)^{\frac{1}{m-1}}}{\sum_{k=1}^{C}\left(\frac{1}{d(W_i,C_k)}\right)^{\frac{1}{m-1}}} \tag{1.7}$$

denoting C_k, the k^{th} class medoid, $d(W_i, C_k)$, the dissimilarity between the term vector W_i and the medoid C_k, and m, a fuzzyfier factor, $m \in [1, \infty)$, denoting the smoothness of the clustering solution ($m = 2$ in this work). The procedure is iterated until either the updated cluster centroids remain the same, or a maximum number of iterations is reached.

Utterance feature vector Finally, the feature vector obtained with soft term clustering, F_{soft}, is calculated as the normalised matrix product between the original bag of words BW and the membership matrix M:

$$F_{soft} = \frac{BW_{(1xD)} \cdot M_{(DxD')}}{|BW \cdot M|}. \tag{1.8}$$

1.7.5 Utterance categorisation

The objective of utterance categorization is to map an input utterance—represented as bag of words (BW) or feature vector after hard or soft word

9) The cosine distance metric, D_{cos} is defined as the negative of the cosine score, $D_{cos} = 1 - S_{cos}$.

clustering—into one of the N categories, denoted by the N prototypes supplied to the nearest neighbour algorithm. The closeness of an input utterance vector to each one of the prototypes is quantified by means of the inner product between their feature vectors, F_i and F_j:

$$s(F_i, F_j) = F_i \cdot F_j^T. \tag{1.9}$$

1.8 Supervised methods for utterance categorization

In this section, we describe two supervised approaches for utterance categorisation: a probabilistic framework (Naïve Bayes classifier) and a vector model with term weighting. F denotes the number of labelled exampes per category, randomly selected.

1.8.1 Naïve Bayes Classifier

Naïve Bayes is a powerful and yet simple text categorisation algorithm usually reporting adequate performance. It selects the most probable class $\hat{c}$ out of a set C given a test utterance u:

$$\hat{c} = \arg\max_{c \in C} P(c|u) \tag{1.10}$$

This expression cannot be computed directly, but it can be reformulated using the Bayes rule as:

$$\hat{c} = \arg\max_{c \in C} P(c)(u|c). \tag{1.11}$$

By assuming conditional independence of the utterance terms, the Naïve Bayes solution can be expressed as:

$$\hat{c} = \arg\max_{c \in C} P(c) \prod_{w_i \in u} (w_i|c) \tag{1.12}$$

where $P(c)$ denotes the class prior probability estimated from the selected set of labelled samples[10]. To deal with the zero probability phenomenon, we applied Laplacian smoothing.

10) The generic variable F is used to reflect the number of examples per category randomly selected from a corpus of labelled utterances.

1.8.2 Vector model with term weighting for utterance classification

In information retrieval, document classification and text summarisation, each document is usually represented by means of a term vector, D (Equation 1.13)

$$D = a_1, a_2, \cdots, a_N \tag{1.13}$$

where the components a reflect the relative significance of terms in relation to the document in hand.

Term scores are generally computed as a contribution of two factors: (i) the absolute or relative frequencies of terms in the document, and (ii) the term dispersion over all documents. The second factor is also used for feature selection characterising the "noisy" behaviour of terms.

Term frequency In the literature, one finds different definitions for the term frequency. In this work, we use two formulations taken from [12] and [32]:

$$TF_1(w,d) = \frac{C(w_i,d)}{\sum_j C(w_j,d)} \tag{1.14}$$

$$TF_2(w,d) = \begin{cases} 1 + log(C(w_i,d)) & \text{if } C(w_i,d) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{1.15}$$

where $C(w,d)$ denotes the occurrence counts of the term w in the document d.

IDF, RIDF and ISCF scores We analysed three relevance scores to capture the term distribution across documents: the inverse document frequency (IDF), residual inverse document frequency (RIDF) and a new formulation, the inverse spectral crest factor (ISCF).

- **Inverse document frequency (IDF).** This popular definition was proposed by [39]:

  $$IDF(w) = -log(\frac{ND_w}{ND}) \tag{1.16}$$

  where ND_w denotes the number of documents in which the term w occurs; and ND is the total amount of documents in the collection. In this work, the number of documents corresponds to the number of categories $ND = L$.

However, the practical number of sample utterances in a given class may be lower than F if there are less than F labelled utterances available for that category. We use this information for the estimation of the category priors $P(c)$.

- **Residual inverse document frequency (RIDF).** This is a variant of the inverse document frequency, proven effective for automatic text summarisation [39]. It represents the difference between the IDF of a term and its expected value $\widehat{IDF}$ according to a Poisson model.

$$RIDF(w) = IDF - \widehat{IDF} \tag{1.17}$$

with

$$\widehat{IDF}(w) = -log(1 - e^{-\lambda_w}) \tag{1.18}$$

where λ_w denotes the parameter of the Poisson distribution, calculated here as the average occurrence of the w term across all ND documents:

$$\lambda_w = \sum_j N_{w_j}/ND. \tag{1.19}$$

Main advantage of $RIDF$ compared to IDF is that rare terms are not assigned exaggerated relevances.

- **Inverse spectral crest factor (SCF).** We propose a third metric called inverse spectral crest factor (ISCF). Motivation for the introduction of this formulation is to achieve a more accurate indicator of the term distribution over the categories. An IDF-based metric would place lower relevance on terms observed in more than one category. However, this metric does not reflect the possibility that terms may occasionally appear in several categories.

  The Spectral Crest Factor (SCF) is one of the measures used in audio processing [35] for determining the noisy character of the signal components through an analysis of their short time spectra. It provides an estimate of the *spectral flatness*, as the ratio of the arithmetic mean energy across spectral bands with respect to the maximum energy. We adopted this metric to estimate a term's dispersion across categories. The term relevance is given by the *inverse spectral crest factor*, defined as:

$$ISCF(w) = \frac{ND \cdot max_i(TF_1(w, d_i))}{\sum_j TF_1(w, d_j)}. \tag{1.20}$$

1.9 Evaluation methods and results

In this section, we describe the methods to evaluate the performance of utterance classification models described in previous sections.

Tab. 1.3 Utterance categorisation with one labelled utterance per class using several feature extraction techniques and disambiguation

Term clustering	Disambiguation	Accuracy
–	no	45%
–	yes	57%
Soft (PoBOC)	no	50%
Soft (PoBOC + FCMdd)	no	47%
Hard	no	50.8%
Hard	yes	62.2%

This is done by comparing the output categories the proposed algorithm assigns to a number of test utterances with manually assigned categories thereof (the reference). If both categories coincide, the automatic categorisation is considered correct, otherwise it is counted as error. As overall accuracy, we define

$$\text{accuracy} = \frac{\#\text{ correctly classified test utterances}}{\#\text{ total utterances in test set}}. \tag{1.21}$$

1.9.1 Classification with one labelled utterance and feature extraction

Table 1.3 shows the accuracy values reached on the Internet corpus by the nearest neighbour classifier applied to bags of words and feature vectors in case of feature extraction, with one sample utterance per category. In this case, the samples have been manually selected in such a way that overlapping terms in different category samples is minimised.

Comparing classification performance without disamgiguation to the baseline (no term clustering; at 45%), we see that both soft and hard term clustering perform very similar: PoBOC and hard term clustering achieve around 50% outperforming the baseline by about 10% relative.

As motivated in Section 1.7.2, disambiguation partially overcame the sparseness of having only one example utterance per class shown by significant improvements from 45% to 57% on the baseline without term clustering (27% relative) and 50.8% to 62.2% on hard term clustering (22% relative). Hard term clustering with disambiguation outperformed the baseline by 38.2% relative.

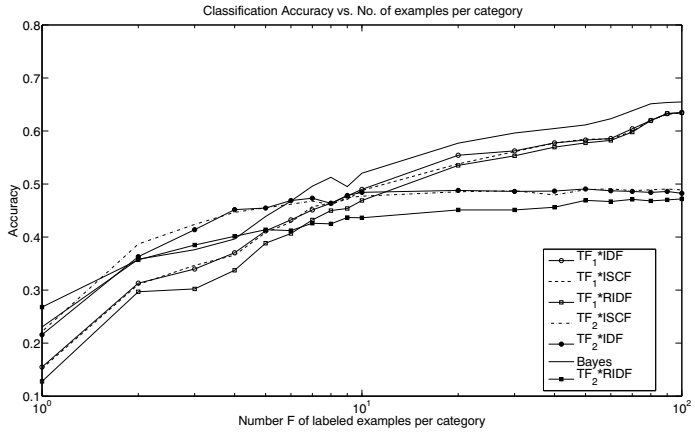

Fig. 1.4 Mean accuracy rates achieved by the Naïve Bayes classifier and a vector model (nearest neighbour) with TFIDF, TFRIDF and TFISCF term weights in a logarithmic x-axis. Reported accuracy values refer to averaged results across 20 runs of the algorithm with different input sets of training utterances, randomly selected from a labelled corpus.

1.9.2 Classification based on F samples per category

The following paragraphs show a comparative analysis of the Naïve Bayes classifier and the approach based on weighted document vectors. In particular, we investigate the dependency between classifier performance and number of (randomly chosen) samples per category ($F \in [1, 100]$). Tests are performed on the Cable TV troubleshooting corpus (10000 test utterances and 79 problem categories). Figure 1.4 depicts the performance of the Naïve Bayes classifier and Nearest Neigbour using term weighting against the number of samples/category F. Based on these experimental results, several observations can be made:

- Naïve Bayes outperforms NN with term weighting and term relevance scoring ($TF_2(w, d)$) for numbers of samples greater than 7. The worse performance of Naïve Bayes in these cases may be attributed to the use of Laplacian smoothing. For small numbers of examples, the ratio of terms with a frequency of zero in the set of examples is rather large (Figure 1.5).

  Therefore, using Laplacian smoothing in conjunction with the Naïve Bayes classifier may produce inaccurate term probability estimates.

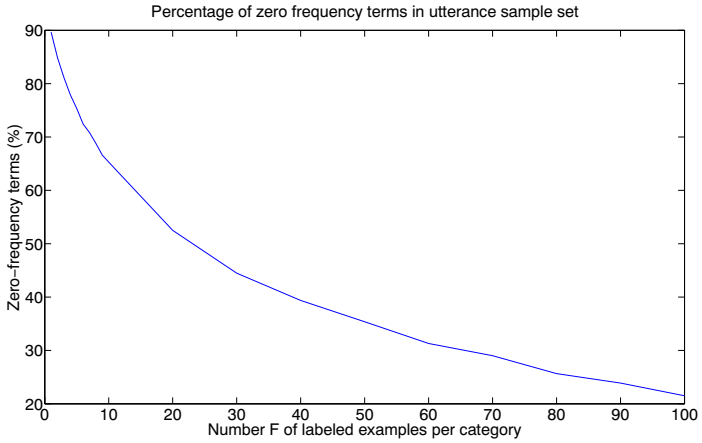

Fig. 1.5 Ratio of terms in test utterances with zero frequency in sample utterance set vs number of sample utterances per category.

Note that, without the use of Laplacian smoothing, the $TF_2(w|c)$ and $P(w|c)$ would be identical.

- We also observed a dependency of classifier's performance on the specific term frequency metric (TF_1 or TF_2, respectively). The normalisation with respect to the document lengths introduced in TF_2 seems to be a better strategy for few examples, but the classifier performance is stabilised after a number $F = 7$ of samples. Also at this point, TF_1 starts to outperform TF_2. One possible reason for this phenomenon is the high sentitivity of classifiers to different utterance lengths when a small number of examples is provided.

- Contribution of IDF, $RIDF$ and $ISCF$: Although $TFRIDF$ was proposed as a more efficient solution in automatic text summarisation, $TFIDF$ has outperformed $TFRIDF$ on this kind of data. This fact may be associated to certain characteristics of the utterance corpus and the way category documents are generated. On the one hand, there is a large number of terms which may be indicative of more than one category. This happens because the mentioned categories indicate different problems which can be experienced with a single device. For example, for problems related to the quality of received image, utterances like *picture has poor quality* are commonly observed, and less frequently, utterances like *bad picture*. However, there also exist other categories to cover additional problems related to the picture. Here, we refer to terms such as *quality*, *poor* or

bad, as specific category terms, in contrast to generic terms like *picture*. A generic term is descriptive for several categories simultaneoulsy, in which it occurs whith considerably high frequencies. Generic terms may be found to the extent that some underlying hierarchical category structure can be assumed. We also distinguish a third kind of terms, refered to as *noisy terms*, which can be observed in many different categories, generally with low frequencies. It is desirable to emphasize specific terms with respect to generic terms, in order to "protect" utterances with a high probability of error like *bad picture*.

In this respect, IDF scores capture a term's spreading over documents regardless of the term frequency in the documents. However, the average frequency of these terms (parameter λ_w of the Poisson model) considerably exceeds that of specific terms. According to a Poisson model, these terms (*picture*) should spread even more over documents in contrast to terms with low λ_w (specific terms), and, therefore, a part of the bias introduced by IDF appears compensated in the residual after subtracting the Poisson estimation $\widehat{IDF}$. Moreover, no significant differences can be observed between $TFISCF$ and $TFIDF$. The use of $ISCF$ scores was motivated to provide a more precise estimate of the term/category distribution which reflects the different frequences of the term in the category documents. However, one fact to be considered is that IDF and $ISCF$ scores are here multiplied to TF scores. This may also explain why, despite of its simplicity, $TFIDF$ scores are among the most broadly used metrics in text processing. Whether $ISCF$ can be effective for global feature selection remains an open question.

1.10 Summary and conclusion

In this article, we described different models for the categorisation of caller utterances in the scope of automated troubleshooting agents . One of the goals of this research is to help overcome costs associated to manual compilation of large training data sets. In the first part of the article, we proposed categorisation schemes which make use of only one labelled sample per category. The proposed solution is based on feature extraction techniques which automatically identify semantic word classes on a corpus of unlabelled utterances. Hard and fuzzy word clustering methods were compared. The performance of feature extraction for utterance classification was experimentally evaluated on a test corpus of more than 3000 utterances and 28 classes. The most optimistic outcomes were achieved with hard word clustering in combination

with a module for reallocating ambiguous utterances providing a maximum of 62.2% accuracy.

The second part of the paper provided an overview of supervised classifiers commonly used for the categorisation of texts. A probabilistic framework (Naïve Bayes) and a vector model with term relevance scores were described. We experimentally compared these classifiers on a test corpus of 10000 utterances and 79 classes. An analysis of the classifier's dependency on the number of labelled examples was carried out. Our experiments reported an inflection point in the classifier's behaviour around seven training samples per category. For lower numbers of training samples, Nearest Neighbor classification with term weighting schemes achieved higher accuracies, whereas for larger numbers, Naïve Bayes outperfoms the other classifiers.

References

1 Acomb K., Bloom J., Dayanidhi K., Hunter P., Krogh P., Levin E., and Pieraccini R. (2007) Technical Support Dialog Systems: Issues, Problems, and Solutions. In *Proc. of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*. Rochester, USA.

2 Agrawal R., Gehrke J., Gunopulos D., and Raghavan P. (1998) Automatic subspace clustering of high dimensional data for data mining applications. pages 94–105.

3 Aksoy S. and Haralick R. (2001) Feature Normalization and Likelihood-Based Similarity Measures for Image Retrieval. *Pattern Recognition Letters* 22(5).

4 Bach E. and Harms R. (1968) *Universals in Linguistic Theory*. Holt, Rinehart and Winston, New York, USA.

5 Blum A. and Mitchell T. (1998) Combining Labelled and Unlabelled Data with Co-Training. In *Proc. of the COLT*. Madison, USA.

6 Bühler D., Minker W., and Elciyanti A. (2005) Using Language Modelling to Integrate Speech Recognition with a Flat Semantic Analysis. In *Proc. of the SIGdial Workshop on Discourse and Dialogue*. Lisbon, Portugal.

7 Buckley C. (1985) Implementation of the SMART information retrieval system. Technical report, Cornell University, Ithaca, USA.

8 Cleuziou G., Martin L., and Vrain C. (2004) PoBOC: An Overlapping Clustering Algorithm. Application to Rule-Based Classication and Textual Data. In *Proc. of the ECAI*. Valencia, Spain.

9 Chapelle O., Schölkopf B., and Zien A. (2006) *Semi-Supervised Learning*. MIT Press, Cambridge, USA.

10 Deerwester S. C., Dumais S. T., Landauer T. K., Furnas G. W., and Harshman R. A. (1990) Indexing by latent semantic analysis. *Journal of the American Society of Information Science* 41(6): 391–407.

11 Dempster A., Laird N., and Rubin D. (1977) Maximum Likelihood from Incomplete Data via EM Algorithm. *Journal of Royal Statistical Society* 39(1).

12 Debole F. and Sebastiani F. (2003) Supervised Term Weighting for Automated Text Categorization. In *Proc. of the SAC*. Melbourne, USA.

13 Ester M., Kriegel H., S J., and Xu X. (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of KDD-96*.

14 Evanini K., Suendermann D., and Pieraccini R. (2007) Call Classification for Automated Troubleshooting on Large Corpora. In *Proc. of the ASRU*. Kyoto, Japan.

15 Hinneburg A. and Keim D. A. (1998) An efficient approach to clustering in large multimedia databases with noise. In *Knowledge Discovery and Data Mining*, pages 58–65.

16 Hofmann T. (1999) Probabilistic latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence, UAI'99*. Stockholm.

17 Hartigan J. and Wong M. (1979) Algorithm AS136: A k-means clustering algorithm. *Applied Statistics* 28: 100–108.

18 (1995) Interactive Services Design Guidelines. Technical Report ITU-T Recommendation F.902, ITU, Geneva, Switzerland.

19 Jain A. and Dubes R. (1988) *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, USA.

20 Jain A. and Mao J. (2000) Statistical Pattern Recognition: A Review. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 22(1).

21 Johnson S. (1967) Hierarchical Clustering Schemes. *Psychometrika* 32(3).

22 Krishnapuram R., Joshi A., Nasraoui O., , and Yi L. (2001) Low-Complexity Fuzzy Relational Clustering Algorithms for Web Mining. *IEEE Trans. on Fuzzy Systems* 9(4).

23 Klein D. and Manning C.-D. (2003) Fast Exact Inference with a Factored Model for Natural Language Parsing. *Advances in Neural Information Processing Systems* 15: 3–10.

24 Kauffmann L. and Rousseeuv P. (1990) *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley & Sons, New York, USA.

25 Li Y. and Jain A. (1998) Classification of Text Documents. *Computer Journal* 41(8).

26 Li Y. H. and Jain A. K. (1998) Classification of Text Documents. *The Computer Journal* 41(8).

27 Levin E. and Pieraccini R. (1995) CHRONUS, the Next Generation. In *Proc. of the ARPA Workshop on Human Language Technology*. Austin, USA.

28 Minker W., Albalate A., Bühler D., Pittermann A., Pittermann J., Strauss P., and Zaykovskiy D. (2006) Recent Trends in Spoken Language Dialogue Systems. In *Proc. of the ICIT*. Cairo, Egypt.

29 M. Ankerst M. M. Breunig H.-P. K. and Sander J. (1999) OPTICS: Ordering Points To Identify the Clustering Structure. In *Proc. of ACM-SIGMOD International Conference on Management of Data*. Philadelphia, Pennsylvania, United States.

30 Minnen G., Carrol J., and Pearce D. (2001) Applied Morphological Processing of English. *Natural Language Engineering* 7(3).

31 Minker W. (1998) *Speech Understanding for Spoken Language Systems – Portability Across Domains and Languages*. Hänsel-Hohenhausen, Frankfurt, Germany.

32 Mori T., Kikuchi M., and Yoshida K. (2001) Term Weighting Method Based on Information Gain Ratio for Summarizing Documents Retrieved by IR Systems. In *Proc. of the NTCIR Workshop*. Tokyo, Japan.

33 Morimoto T., Shikano K., Iida H., and Kurematsu A. (1990) Integration of Speech Recognition and Language Processing in Spoken Language Translation System (SL-Trans). In *Proc. of the ICSLP*. Kobe, Japan.

34 Ng R. and Han J. (1994) Efficient and effective clustering methods for spatial data mining. In *Proc. of the 20th Conference on VLDB*. Santiago, Chile.

35 Peeters G. (2003) A Large Set of Audio Features for Sound Description. Technical report, IRCAM, Paris, France.

36 Picard J. (1999) Finding Content-Bearing Terms using Term Similarities. In *Proc. of the EACL*. Bergen, Norway.

37 Pyle D. (1999) *Data Preparation for Data Mining*. Morgan Kaufmann, Los Altos, USA.

38 Rabiner L. (1989) A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. of the IEEE* 77(2).

39 Renals G. M. S. (2007) Towards Online Speech Summarization. In *Proc. of the Interspeech*. Antwerp, Belgium.

40 Salton G., Wong A., and Yang C. S. (1975) A Vector Space Model for Automatic Indexing. *Communication of the ACM* 18(11).

41 Wagstaff K., Cardie C., Rogers S., and Schroedl S. (2001) Constrained K-Means Clustering with Background Knowledge. In *Proc. of the ICML*. Williamstown, USA.

42 Wulfekuhler M. and Punch W. (1997) Finding Salient Features for Personal Web Page Categories. In *Proc. of the International Web Conference*. Santa Clara, USA.

43 Zhang T., Ramakrishnan R., and Livny M. (1997) Birch: A New Data Clustering Algorithm and Its Applications. *Data Mining and Knowledge Discovery* 1(2).