Sketch and Attribute Based Query Interfaces

by

Caglar Tirkaz

Submitted to the Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer Science
at the
SABANCI UNIVERSITY
June 2015

Author .....
Computer Science and Engineering
April 29, 2015

Approved by .....
A. Berrin Yanıkoğlu
Associate Professor
Thesis Supervisor

Approved by .....
T. Metin Sezgin
Assistant Professor
Thesis Supervisor

Approved by .....
Hakan Erdoğan
Associate Professor

Approved by .....
Kamer Kaya
Assistant Professor

Approved by .....
Tolga Taşdizen
Associate Professor

APPROVE DATE: .....

Sketch and Attribute Based Query Interfaces

by

Caglar Tirkaz

Submitted to the Computer Science and Engineering
on April 29, 2015, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science

Abstract

In this thesis, machine learning algorithms to improve human computer interaction are designed. The two areas of interest are (i) sketched symbol recognition and (ii) object recognition from images. Specifically, auto-completion of sketched symbols and attribute-centric recognition of objects from images are the main focus of this thesis. In the former task, the aim is to be able to recognize partially drawn symbols before they are fully completed. Auto-completion during sketching is desirable since it eliminates the need for the user to draw symbols in their entirety if they can be recognized while they are partially drawn. It can thus be used to increase the sketching throughput; to facilitate sketching by offering possible alternatives to the user; and to reduce user-originated errors by providing continuous feedback. The latter task, allows machine learning algorithms to describe objects with visual attributes such as "square", "metallic" and "red". Attributes as intermediate representations can be used to create systems with human interpretable image indexes, zero-shot learning capability where only textual descriptions are available or capability to annotate images with textual descriptions.

Thesis Supervisor: A. Berrin Yanıkoğlu
Title: Associate Professor

Thesis Supervisor: T. Metin Sezgin
Title: Assistant Professor

Çizim ve özellik temelli sorgu arayüzleri

Çağlar Tırkaz

tarafından
Bilgisayar Bilimi ve Mühendisliği departmanına
29 Nisan 2015 tarihinde,
Bilgisayar Bilimi Doktorası
gereksinimleri için teslim edilmiştir

Özet

Bu tezde insan bilgisayar etkileşimini gelişterecek makine öğrenmesi metotları tasarlanmıştır. Tezin ilgilendiği iki konu (i) çizilmiş sembol tanıma ve (ii) resimlerden obje tanımadır. Spesifik olarak ise, tamamı çizilmeden sembollerin tanınması ve özellik tanıma kullanılarak resimlerdeki objelerin tanınması tazin temel çalışma alanını oluşturmaktadır. Tamamlanmamış sembollerin tanınmasının kullanıcının çizim yapma hızını arttırmak, kullanıcıya geri bildirim sağlamak, ya da kullanıcı hatalarını azaltmak gibi birçok kullanım alanı bulunmaktadır. Özellik tanıma ise resimleri insanların objeleri tanımlamak için kullandığı "kare", "metalik" ya da "kırmızı" gibi görsel özelliklerle açıklamayı sağlar. Objelerin özellikleri, resimlerin insanlar tarafından anlaşılabilen kelimelerle indekslenmesi, daha önce görülmemiş sınıfların sadece objeleri anlatan açıklamalar vasıtasıyla tanınması veya resimleri açıklayan yazılar üretilmesi gibi birçok alanda kullanılabilir.

Tez Danışmanı: A. Berrin Yanıkoğlu
Ünvanı: Doçent Doktor

Tez Danışmanı: T. Metin Sezgin
Ünvanı: Yardımcı Doçent

Teşekkür

Doktoram süresince bana yol gösteren ve araştırmalarıma yön veren danışman hocalarım Berrin Yanıkoğlu ve Metin Sezgin'e; Yaklaşık 9 ayımı geçirdiğim Georgia Tech. üniversitesinde danışmanım olan Jacob Eisenstein'a; Benden desteklerini esirgemeyen aileme; Her zaman yanımda olan, bana moral ve enerji veren, yola devam etmemi sağlayan Duygu Koç'a; Sağladığı imkanlarla Türkiye'nin en iyi üniversitelerinden biri olan Sabancı Üniversitesi'ne ve üniversiteyi yaratan Sabancı ailesine; Amerika'da doktora çalışması yapmam için burs sağlayan Fulbright'a; Son olarak doktara sürecimde 2211 Yurt İçi Lisansüstü Burs Programı kapsamında beni destekleyen TÜBİTAK'a teşekkür ederim.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning (ML) research field deals with the design and study of algorithms for enabling machines to learn, for understanding, modeling and making decisions from data. ML techniques are employed in various domains such as computer vision, natural language processing, robotics, bioinformatics and information retrieval. Over the past decade ML has seen the development of faster and more accurate algorithms that can be applied under a broader array of problem structures and constraints.

Human-computer interaction (HCI), in contrast, is concerned with the human context focusing on the interfaces between people and computers. HCI research puts a significant emphasis on developing technology and practices that improve the usability of computing systems, where usability encompasses the effectiveness, efficiency, and satisfaction with which the user interacts with a system.

The amount of data coming from diverse sources is increasing at an exponential rate. People deal with the growing data which increase the need to extract meaningful information from and manage accumulating data. Consider browsing through an image collection of thousands of images when searching for a specific image. It might be really time-consuming to find what you are looking for. However, if you could have a way to describe the image you are looking for and the computer retrieved only the relevant images then the time you spent would be reduced drastically. As an another example, consider an interface where sketching is used as the medium of interaction with many symbols. When a user wants to sketch a symbol it might be hard for

the user to remember the symbol correctly in its entirety. However, the user might remember parts of the symbol and it would be nice if the computer could understand what the user intends to draw and make suggestions to the user as to what s/he wants to draw. This thesis is in the crossroads of ML and HCI. We introduce novel computer vision and machine learning algorithms to improve and facilitate human computer interaction, especially when working with large collections.

In Chapter 2 we work on sketched symbol recognition and we introduce and evaluate a method for auto-completion of sketched symbols (The methods discussed in Chapter 2 are published in [74]). Sketching is a natural mode of communication amongst humans that is especially useful in some domains such as engineering and design. It can be used to convey ideas that are otherwise hard to describe verbally. Increasing availability of touch screens makes sketching more viable than ever and increases the need to create user-friendly sketching interfaces. In order to create such interfaces we need computer applications that can "understand" what the user intends to draw and interact with the user in a natural way. Sketch recognition aims to provide users with such applications where the input is hand-drawn symbols with the output being the recognized symbols. Sketch recognition is a well studied field and approaches to sketch recognition such as gesture-based, rule-based and image-based are proposed in the literature. However, while the prior work in sketch recognition focuses on recognition of symbols or scenes in their entirety, we aim to improve the sketching interfaces by providing auto-completion capability. Auto-completion allows a user to complete a sketched symbol even before drawing it entirely. Auto-completion can be useful in various ways such as improving the speed of the user, beautification of sketches, helping the user remember symbols and preventing user errors through feedback. These features enable a more intuitive and user friendly experience and improve user satisfaction.

In Chapter 3 attribute-centric recognition of objects in images is studied (The methods discussed in Chapter 3 are published in [73]). The performance of algorithms designed to recognize object categories from images is increasing each year. Most image classification approaches rely on level features such as SIFT, HOG and

SURF to train classifiers to discriminate object categories. More recently, algorithms based on deep-learning achieved significant improvements in image classification tasks. Deep-learning algorithms rely on the availability of massive annotated datasets and computing power. However, such annotated data might not be available for the recognition task at hand. One alternate approach is to describe object categories using the visual attributes they have in order to create an intermediate representation. Attributes encode shared visual traits among categories such as square, furry, metallic, animal and vehicle. Having such an intermediate representation not only makes it easier to create models with a limited amount of data but also allows for a human understandable way of describing images. Through attributes, it is possible to create a system that is able to describe an image of a category (*e.g.*, image of a horse) with the existing attributes in the image (*e.g.*, four-legged, animal, mammal) without seeing an image of the described category (*e.g.*, horse). In the thesis, we introduce and evaluate methods to automatically mine candidate attributes that describe objects visually. Attributes have attracted a lot of interest recently and they have been proven useful in various recognition tasks. However, the question of how to select the attributes and how to find category-attribute associations remains relatively untouched. We show that a taxonomy over object categories can be leveraged to automatically mine attributes from textual descriptions of the categories. The mined attributes are valuable to improve HCI since they are meaningful to humans and they can be leveraged to create user friendly interfaces.

In Chapter 4, we introduce interfaces arising from the ideas developed int Chapter 2 and Chapter 3. Specifically, an interface with auto-completion support to recognize sketched symbols and another one with the ability search through images using attributes are presented. The interfaces we implement illustrate two of the many ways auto-completion and attributes can be utilized and how HCI can benefit from them.

Finally in Chapter 5, we discuss how the work in this thesis can be further improved and future research directions.

Chapter 2

Sketched symbol recognition with auto-completion

Sketching is a natural mode of communication that can be used to support communication among humans. Recently there has been a growing interest in sketch recognition technologies for facilitating human-computer interaction in a variety of settings, including design, art, and teaching. Automatic sketch recognition is a challenging problem due to the variability in hand drawings, the variation in the order of strokes, and the similarity of symbol classes. In the thesis, we focus on a more difficult task, namely the task of classifying sketched symbols before they are fully completed. There are two main challenges in recognizing partially drawn symbols. The first is deciding when a partial drawing contains sufficient information for recognizing it unambiguously among other visually similar classes in the domain. The second challenge is classifying the partial drawings correctly with this partial information. We describe a *sketch auto-completion* framework that addresses these challenges by learning visual appearances of partial drawings through semi-supervised clustering, followed by a supervised classification step that determines object classes. Our evaluation results show that, despite the inherent ambiguity in classifying partially drawn symbols, we achieve promising auto-completion accuracies for partial drawings. Furthermore, our results for full symbols match/surpass existing methods on full object recognition accuracies reported in the literature. Finally, our design allows real-time

symbol classification, making our system applicable in real world applications.

2.1 Motivation

Sketching is the freehand drawing of shapes and is a natural modality for describing ideas. Sketching is of high utility, because some phenomena can be explained much better using graphical diagrams especially in the fields of education, engineering and design. Sketch recognition refers to recognition of pre-defined symbols (e.g. a resistor, transistor) or free-form drawings (e.g. an unconstrained circuit drawing); in the latter case, the recognition task is generally preceded by segmentation in order to locate individual symbols. There are many approaches in the literature for sketched symbol recognition. These include gesture-based approaches that treat the input as a time-evolving trajectory [66, 49, 82], image-based approaches that rely only on image statistics (e.g., intensities, edges) [40, 36, 55], or geometry-based approaches that attempt to describe objects as geometric primitives satisfying certain geometric and spatial constraints [33, 34, 13]. However, these methods mostly focus on recognizing fully completed symbols. In contrast, here we focus on the recognition of partially drawn symbols using image-based features.

The term auto-completion refers to predicting the sketched symbol before the drawing is completed, whenever possible. Auto-completion during sketching is desirable since it eliminates the need for the user to draw symbols in their entirety if they can be recognized while they are partially drawn. It can thus be used to increase the sketching throughput; to facilitate sketching by offering possible alternatives to the user; and to reduce user-originated errors by providing continuous feedback [10]. Despite these advantages, providing continuous feedback might also distract the user if premature recognition results are displayed [25, 40].

Auto-completion requires continuously monitoring the user's drawing and deciding whether the input given thus far can be recognized unambiguously. In order to formalize the terms ambiguity and confidence, consider the task of auto-completion in SMS applications where the task is to try to guess the intended word before it is

(a) (b)

Figure 2-1: Two sample sketched symbols from the COAD database.

completely typed, so as to increase typing throughput. For this problem, suppose the language consists of three words: *cat, car,* and *apple*. If the first input character is 'a', then the word auto-completion system can infer the intended word ("apple") unambiguously. On the other hand, if the first character is 'c' and no other information is available about the language, the intended word is ambiguous (either "cat" or "car") and a text-based auto-completion system can be only 50% confident. However, suppose that the same auto-completion system is allowed to make 2 guesses on the word the user intends to type. Then, the system can guess the top 2 choices as "car" and "cat" with 100% confidence as no ambiguity is present.

Sketch recognition is a difficult problem due to the variability of user's hand drawing, the variability in the stroke order and the similarity of sketch classes to be recognized. Sketch recognition with auto-completion is further complicated since the system is faced with the problem of computing a confidence during the recognition process. A hand-drawn symbol is ambiguous if it appears as a sub-symbol of more than one symbol class. This is often the case with partial symbols and occasionally even with fully completed symbols.

Note that in the auto-completion framework, the system is not told when the drawing of a symbol ends. This introduces additional difficulty in classifying *full* symbols as well. For example, although the symbol shown in Figure 2-1a is a fully completed symbol, it appears as a sub-symbol of another symbol shown in Figure 2-1b. Hence, without knowing that the drawing of a symbol ended, a symbol such as the one shown in Figure 2-1a would be classified as ambiguous. The issue of the ambiguity of fully completed symbols is discussed further in Section 2.3.2.

Supplying the user with predictive feedback is an important problem that has been previously studied (in terms of its effects, desirable extent etc.) [1, 78]. Most of

the previous work has focused on giving this feedback in the form of beautification. In the context of sketch recognition, the word 'beautification' has been used in two different senses. First, it refers to recognizing and replacing a fully completed symbol with its cleaned-up version [63, 2, 37] Second, it is used in the context of partial drawings to refer to converting the strokes of a symbol to vectorized primitives such as line segments, arcs, and ellipses [59, 60, 46] Sometimes these primitives are further processed to adhere to Gestalt principles (e.g., lines that look roughly parallel/equal-length are made parallel/equal-length) [69, 61, 38] Approaches of the first kind are not directly comparable to our work, as they only deal with fully completed symbols. Approaches of the second kind are also not very relevant in the context of our work, because in these systems the primitives are recognized and post-processed using Gestalt principles, however the object class is not predicted.

An implementation of beautification that couples with the idea of auto-completion has been proposed by Arvo and Novins [4]. They introduce the concept of *fluid sketching* for predicting the shape of a partially drawn primitive (e.g., a circle or a square), as it is being drawn. However, they focus on primitives only, and don't generalize their system to recognize complex objects. Li *et al.* [47] use the term *incremental intention extraction* to describe a system that can assist the user with continuous visual feedback. This method also has the ability to update existing decisions based on continuous user input. They focus on recognizing multi-lines and elliptic arcs. Mas *et al.* [51] present a syntactic approach to online recognition of sketched symbols. The symbols are defined by an adjacency grammar whose rules are generated automatically given the small set of 7 symbols. The system can recognize partial sketches in arbitrary drawing order, using the grammar to check the validity of its hypotheses. The main shortcoming of this system is its syntactic approach, consisting of rigid rules for rule application and primitive recognition. In comparison, we use image features to describe individual symbols to handle different drawing orders and our framework is fully probabilistic.

An auto-completion application similar to ours deals with the auto-completion of complex Chinese characters in handwriting recognition, in which the auto-completion

is used to facilitate the input by providing possible endings for a given partial drawing. For instance, Liu *et al.* [48] use a multi-path HMM to model different stroke orders that may be seen in the drawing of a character. They report accuracies with respect to the percentage of the whole character trajectory written. They obtain accuracies of 82% and 57% when 90% and 70% of the whole character is drawn, respectively

In the thesis, we present a general auto-completion application that is capable of auto-completing sketched symbols without making any assumptions about the complexity of symbols or the drawing style of users or the domain. The system classifies sketched symbols into a set of pre-defined categories while providing auto-completion whenever it is confident about its decision. The steps of the proposed method for auto-completion are explained in detail in Section 2.2; the experiments on databases using the method are described in Section 2.3; the results of the experiments are discussed in Section 2.4; and future directions for research are presented in Section 2.5.

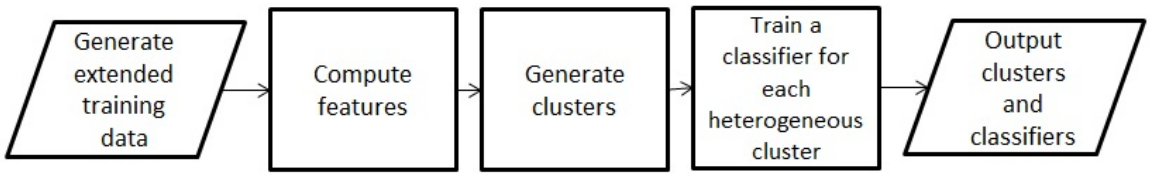
2.2 Proposed method

In order to realize auto-completion, our system monitors the user's drawing and determines probable class labels and assigns a probability to each class as soon as new strokes are drawn. If the drawn (partial or full) symbol can be recognized with a sufficiently high confidence, the system makes a prediction and displays its classification result to the user. Otherwise, classification decision is delayed until further strokes are added to the input symbol.

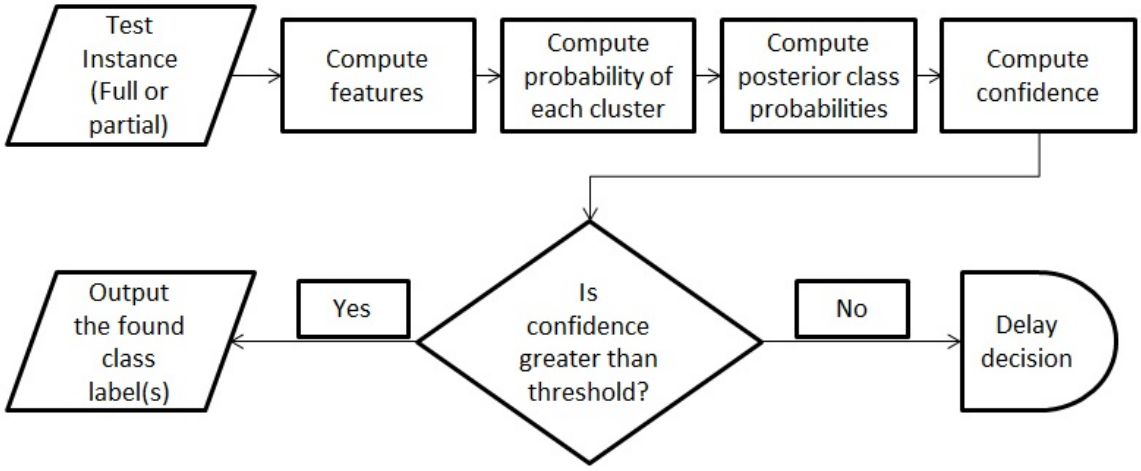
In order to deal with the ambiguity of partial symbols, a constrained semi-supervised clustering method is applied to create clusters in the sketch space. The sketch space is acquired by extracting features from the *extended* training data, which consists of only full symbols and their corresponding partial symbols. Specifically, each full symbol in the training data and all partial symbols that appear during the course of drawing that symbol, are added to extended training data (see Section 2.2.1). The goal of the clustering stage is to identify symbols that are similar based on the extracted features, but may belong to different classes (see Section 2.2.3). At the end

of clustering, a cluster may contain partial/full symbols from only one class (homogeneous cluster) or from multiple classes (heterogeneous cluster). Hence, in the last step of training, we use supervised learning where one classifier per heterogeneous cluster is trained to separate the symbols falling into that cluster (see Section 2.2.4). If a cluster is homogeneous, then a classifier is not needed for that cluster.

During recognition, the system first finds the distance of a symbol to each of the clusters and then computes the posterior probability of each sketch class given the input, by marginalizing over clusters (see Section 2.2.5). This is done so as to take into account the ambiguity in assessing the correct cluster for a given query. Dealing with probabilities allows us to compute a confidence in the classification decision during the test phase. If the class label cannot be deduced with a confidence higher than a pre-determined threshold, as in the case of a partial symbol shared by many classes, the classification decision is postponed until more information becomes available. The described steps are displayed in the form of a flowchart in Figure 2-2.



(a) The training steps



(b) The testing steps

Figure 2-2: The flowchart of the proposed algorithm

(a) After first stroke (b) After second stroke (c) After third stroke (d) Fully completed symbol

Figure 2-3: A sample of extending an instance with four strokes. The original symbol shown in Figure 2-3d, is used to generate the three other symbol instances.

2.2.1 Extending training data with partial symbols

In standard sketched symbol databases, there are only instances of fully completed symbols rather than partial symbols. In our approach, the training and test data are *automatically* extended by adding all the partial symbols that occur during the drawing of fully completed symbols. More specifically, if a particular symbol consists of three strokes, (s_1, s_2, s_3), two partial symbols are extracted $\{(s_1), (s_1, s_2)\}$ and added to the database. For another user who draws the same symbol using the order, (s_2, s_1, s_3), the partials $\{(s_2), (s_2, s_1)\}$ are extracted. In this fashion, for a symbol that consists of S strokes, $S-1$ partial symbols are extracted and added to the extended database, in addition to the original symbol. This process is illustrated in Figure 2-3.

The number of all partial symbols that can be generated using S strokes is exponential in the number of strokes if all combinations of strokes are used. In other words, if we disregard the order between the strokes, we get $2^S - 1$ possible stroke subsets for a symbol with S strokes. However, since we extend the database with only those partials that actually appear in the drawing of the symbols, the number of partial symbols added to the database is much smaller. This issue can be illustrated with an example of drawing of a stick figure. If no one draws a stick figure starting with the head which is then followed by the left leg, the system would not add a partial symbol consisting of the head and the left leg into the database.

Hence, even though a pre-specified drawing order is *not* required by our system, the system takes advantage of preferred drawing orders, when they exist. Indeed, it is true that when people sketch, they generally prefer a certain order. This is

based on observations from previous work in sketch recognition and psychology, which show that people do tend to prefer certain orderings over others [70, 76, 72]. So, our approach puts the focus on learning the drawing orders that are present in the training data, so as to reduce the complexity of the sketch space and improve accuracy. However, a partial symbol that results from a different drawing order may still be recognized by the system depending on its similarity to the instances in the sketch space.

2.2.2 Feature extraction

In order to represent a symbol, which may be a partial or a full symbol, the Image Deformation Model Features *(IDM)* are used as proposed in [56]. The IDM features consist of pen orientation maps in 4 orientations and an end-point map indicating the end points of the pen trajectory. In order to extract the IDM features for a symbol, firstly, the orientation of the pen trajectory at each sampled point in the symbol is computed. Next, five maps are created to represent the IDM features. The first four maps correspond to orientation angles of 0, 45, 90 and 135 degrees, where each map gives a higher response at locations in which the pen orientation coincides with the map orientation. The last map gives a higher response at end-points where a pen-down or pen-up movement occurs. These operations are carried out using a down sampled version of the symbol. The major advantage of the IDM feature representation is that it is independent of stroke direction and ordering.

2.2.3 Clustering

There is an inherent ambiguity in decision making during auto-completion. In order to address this ambiguity, we cluster partial and full symbols based on their feature representation. Clusters which contain drawings mostly from a single class indicate less ambiguity, whereas clusters that contain drawings from many distinct classes indicate high ambiguity.

In order to cluster training instances, we first experimented with the unsupervised

Expectation Maximization (EM) algorithm [17]. We used the implementation of EM available in *WEKA* [32]. The results with the EM algorithm showed a low performance for full symbols. Since our goal is to provide auto-completion without sacrificing full symbol recognition performance, we switched to the semi-supervised constrained k-means clustering algorithm (*CKMeans*) [77]. Our motivation when using CKMeans is to enforce the separation of full symbols of different classes into different clusters, while grouping the full symbols of the same class in one cluster through constraints. With this approach, we aim to reduce errors in classifying full symbols, since errors done in full symbols may distract the user more than errors done in partial symbols. The effect of the clustering algorithm on the recognition accuracies is further discussed in Section 2.3.6.

The CKMeans algorithm employs background knowledge about the given instances and uses constraints of the form *must-link* and *cannot-link* between individual instances while clustering the data. The must-link constraint between two instances specifies that the two instances should be clustered together, whereas the cannot-link constraint specifies that the two instances must not be clustered together. We generate:

- Must-link constraints between full sketches of a class since we want them to be clustered together.
- Cannot-link constraints between full sketches of different classes since we do not want them to be clustered together.

A visual depiction of the must-link and cannot-link constraints between the fully drawn symbols, is given in Figure 2-4. The must-link constraints are shown as circles, indicating that circled instances should be clustered together; while the cannot-link constraints are shown as crossed lines between circles, indicating that full-shape instances in different classes should not be clustered together. No constraints are generated for partial symbols. We allow partial sketches of different classes to be clustered together because partial sketches of different classes can be visually similar and have similar feature representations.

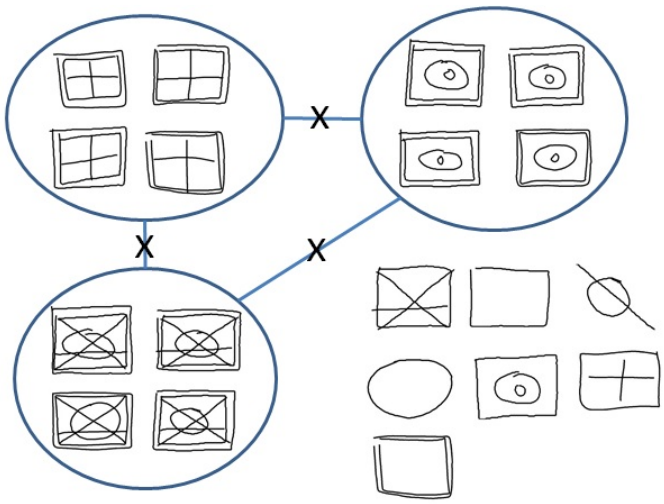


Figure 2-4: A visual depiction of defined constraints used in the CKMeans algorithm. Must-link and cannot-link constraints involving full shapes are represented as circles and crossed lines, respectively.

The constraints are specified using an $N \times N$ symmetric matrix, where N is the number of instances to be clustered in the extended training set and the matrix elements can be -1, 0, 1 denoting cannot-link, no constraint and must-link constraints respectively. The process of generating constraints is handled fully automatically using the class labels present in the original training data.

2.2.4 Posterior class probabilities

In order to make a prediction given a test symbol, x, we compute the posterior probability of each symbol class s_i, by marginalizing over clusters:

$$\begin{aligned} P(s_i|x) &= \sum_{k=1}^{K} P\left(s_i, c_k|x\right) \\ &= \sum_{k=1}^{K} P\left(s_i|c_k, x\right) P\left(c_k|x\right) \end{aligned} \tag{2.1}$$

where x represents the input symbol; K is the total number of clusters; $P\left(s_i|c_k, x\right)$ is the probability of symbol class s_i given cluster c_k and input x; and $P\left(c_k|x\right)$ denotes the posterior probability of cluster c_k given x. Notice that rather than finding the

most likely cluster, we take a Bayesian approach and consider $P(c_k|x)$ in order to reflect the ambiguity in cluster selection.

Given the distance from x to each cluster center, an exponentially decreasing density function and the Bayes' formula, we estimate $P(c_k|x)$ as:

$$\begin{aligned} P(c_k|x) &= P(x|c_k)\, P(c_k)/P(x) \\ &\simeq e^{-||x-\mu_k||^2} P(c_k)/P(x) \end{aligned} \tag{2.2}$$

where μ_k is the mean of the k^{th} cluster c_k and $P(c_k)$ is the prior probability of c_k estimated by dividing the number of instances that fall into the k^{th} cluster by the total number of clustered instances. $P(x)$ denotes the probability of occurrence of the input x, which is omitted in the calculations since it is the same for each cluster.

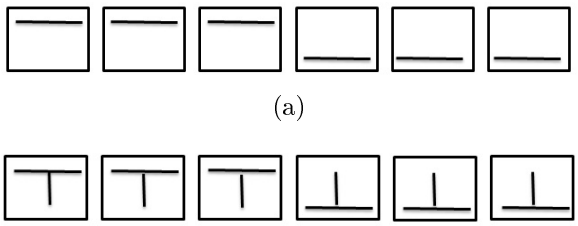
Supervised classification within a cluster

In order to compute $P(s_i|c_k, x)$, a support vector machine (SVM) [15] is trained for each *heterogeneous* cluster, which is defined as a cluster that contains instances of more than one class. If an instance is clustered into a *homogeneous* cluster, which is defined as a cluster containing instances of only a single class, then we simply assign a probability of 1 for the class that forms the cluster and 0 for the other classes.

Note that supervised classification step can help in cases where the symbols falling into one cluster can actually be classified unambiguously. For instance, consider the synthetic cluster given in Figure 2-5a containing six partial symbols. Furthermore, assume that the corresponding fully completed drawings are given in Figure 2-5b. Hence, the partial symbols in the cluster belong to two distinct classes, either the class with an upside 'T' or a downside 'T' inside a square. While the partial symbols in the cluster look similar enough to be clustered together, the position of the line in the square can be used to separate them apart. This is the motivation for training a classifier to separate the instances falling in heterogeneous clusters. If all the symbols that fall in a cluster look very similar, the supervised classification may not bring any contribution and the shapes falling in that cluster would be labeled as ambiguous by

the system, since multiple classes would have similar posterior probabilities.

The semi-supervised clustering step used before the supervised classification performed for each cluster, aims to divide the big problem, into smaller problems that are hopefully easier to solve.



(a)

(b)

Figure 2-5: The visual representation of a synthetic cluster containing 6 symbols is given in Figure 2-5a. The completed drawings are given in Figure 2-5b.

In order to show the contribution of the supervised classification step, we conducted an experiment in which we assumed $P(s_i|c_k, x) = P(s_i|c_k)$ and modified Eq. 2.1 accordingly. Specifically, if it is assumed that the probability $P(s_i|c_k, x)$ is independent of the input instance x, then:

$$P(s_i|c_k, x) = P(s_i|c_k) \tag{2.3}$$

and $P(s_i|c_k)$ can be estimated during training by dividing the number of instances from symbol class i that fall into cluster k by the total number of instances in that specific cluster. Of course, with this assumption there is a loss of information and we see a decrease in the accuracies, as explained in Section 2.3.7.

2.2.5 Confidence calculation

Having computed the posterior class probabilities for an input symbol, the system either rejects the symbol (delays making a decision) or shows the inferred class label(s) to the user. In an auto-completion scenario, the user may be interested in seeing the Top-N guesses of the system and choose from among those to quickly finish drawing his/her partial symbol. For example, if $N = 2$, the system shows the user two alternative guesses.

Naturally, as N increases, the accuracy increases, though too many alternatives would also clutter the user interface. Keeping N as variable that can be set by a user, the confidence in prediction is calculated by summing the estimated posterior probabilities of the most probable Top-N classes. The classification decision is delayed until there is enough information to unambiguously classify the symbol if the computed confidence is lower than a threshold. We refer to the proportion of symbols that are not classified due to low confidence as "reject rate". If the confidence is above the threshold, the N most probable classes are displayed to the user. In the experiments, the system performance is measured for $N = [1, 2, 3]$.

2.3 Experimental results

The proposed system is evaluated on two databases from different domains, in terms of the Top-N classification accuracy in full and partial symbols separately, for varying values of N. For each database, the system parameters (the number of clusters, K, and the confidence threshold, C) are optimized using cross-validation.

Parameter optimization is done as follows: for each parameter value pair (e.g. $K = 40$ and $C = 0.0$), we record the validation set accuracy using 8-fold cross-validation. Cross-validation is done by splitting the training data randomly, selecting 80% of the full symbols and all of their partials as training examples and the remaining 20% of the full symbols and all of their partials as validation examples. This is repeated 8 times with randomly shuffled data and the median system performance on the validation set is recorded, for that particular parameter combination. The selected parameter pair is then fixed and used in testing the system on a separate test set.

2.3.1 Databases

The first database we use to test our system is the Course of Action Diagrams (COAD) database. The COAD symbols are used by military in order to plan field operations [22]. The symbols in this database represent military shapes such as a friendly or

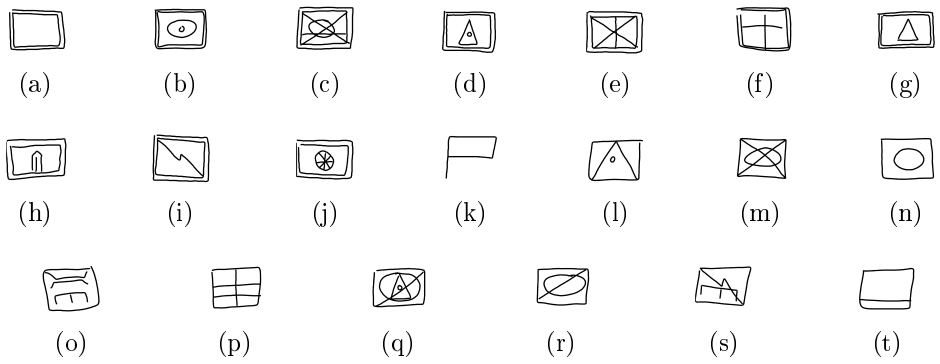


Figure 2-6: A sample symbol from each class in the COAD database.

enemy units, obstacles, supply units, *etc*. Some samples of the hand-drawn symbols from this domain are displayed in Figure 2-6. As mentioned before, some symbols have distinctive shapes whereas others appear as partial symbols of one or more symbols. For example, Figure 2-6n is a sub-shape of Figure 2-6m. In total this database contains 620 samples from 20 symbols drawn by 8 users.

Since no separate test set is available for the COAD database, a randomly selected 20% of all the available data is reserved for testing, prior to parameter optimization done with cross-validation. The parameter optimization for the COAD database aims to find (C, K) pairs at which the system performs close to human recognition rates as will be described in Section 2.3.2. We report the system performance on this test set in detail in Sections 2.3.3 and 2.3.4.

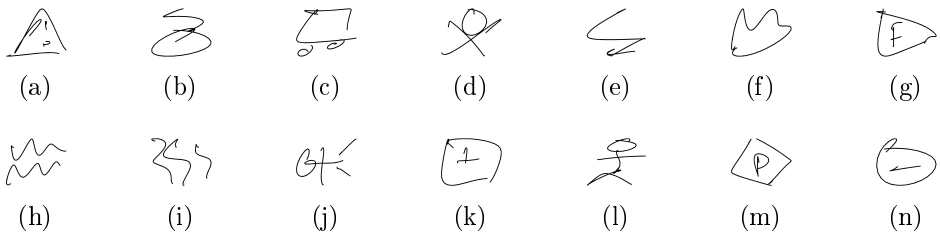


Figure 2-7: A sample symbol from each class in the NicIcon database.

The second database we used in our experiments is the NicIcon [54] symbol database used in the domain of crisis management. The database contains 26163 symbols representing 14 classes collected from 32 individuals. The symbols represent events and objects such as accident, car, fire, *etc*. Some of the sketched symbols from the database are displayed in Figure 2-7. The NicIcon database defines the training

and test sets and in the experiments we used these sets accordingly.

2.3.2 Auto-completion performance benchmark

There are no reported auto-completion accuracies for the COAD and the NicIcon databases, in the literature. However, both databases have been used before in testing sketched symbol recognition algorithms designed for classifying full symbols. While presenting the results of our experiments on the databases, we give both partial and full symbol recognition performances and compare them to full symbol recognition rates from the literature.

In order to test the accuracy decrease due to the auto-completion scenario (since without knowing that the drawing ended, we cannot be sure of the class), we measured a human expert's performance on the COAD database, assuming an auto-completion framework. Specifically, we showed all partial and full symbols in the COAD database to a human expert without telling whether the drawing was finished or not. The expert was then asked to choose the correct class if the the sample could be classified unambiguously for varying values of N, and reject it otherwise.

The first row of Table 2.1 indicates that 75.36% of the partial symbols and 33.58% of full symbols are found to be ambiguous when $N = 1$; that is when the expert is asked to identify the correct class. The symbols that were not rejected were classified with 100% accuracy. As mentioned before, both partial and full symbols in a database may be ambiguous in an auto-completion scenario, without knowing that the user has finished drawing. In particular, full symbols that are found ambiguous are those that can be partial drawings of other symbols.

For $N = 2$ and $N = 3$, the task is to decide whether the symbol can be placed with certainty in one of N possible classes, hence, the reject rates decrease as N increases. Human performance for the NicIcon database was not calculated due to the large size of the database and the amount of manual work involved.

Human recognition rates may be used as a point of reference for assessing an automatic recognition system's performance. In particular, we can compare the proposed system's accuracy to that of the human expert's, at the reject rates close to

Top-N Policy	Partial Accuracy	Full Accuracy	Reject rate for Partial	Reject rate for Full
N=1	100%	100%	75.36%	33.58%
N=2	100%	100%	61.74%	18.25%
N=3	100%	100%	55.07%	12.41%

Table 2.1: Human accuracy showing the proportion of partial and full symbols that need to be rejected in order to achieve 100% accuracy, for varying values of $N = [1-3]$ on the COAD database. The reject rate indicates percentage of the cases where the human expert decided that there is not sufficient information for classification, hence declined prediction.

the expert's.

2.3.3 Accuracy on the COAD database with EM clustering

As described in Section 2.2.3, we first used the EM method for clustering. We summarize the validation and test performances on the COAD database using the EM algorithm, so as to motivate the use of the CKMeans algorithm.

Table 2.2 shows a summary of the cross-validation accuracies obtained with different values for the system parameters (cluster count parameter K and confidence threshold C) that give reject rates close to human reject rates, for comparability. The best parameter pair giving the highest validation set accuracy is indicated with an asterisk. We then used the chosen parameters ($K = 40, C = 0.84$) to evaluate the test set performance, obtaining the results shown in Table 2.3. The human accuracies and reject rates measured on the whole COAD database, are also listed for easy comparison.

K/C	Partial Accuracy	Full Accuracy	Reject Rate for Partial	Reject Rate for Full
80 / 0.87	88.40%	96.90%	71.68%	33.87%
60 / 0.84	91.50%	98.31%	73.26%	33.87%
40 / 0.84*	91.37%	98.44%	73.83%	33.33%

Table 2.2: Validation accuracies for the COAD database for $N = 1$ using the EM algorithm.

While the results shown in Table 2.3 are already good (lower reject rates com-

K/C	Partial Accuracy	Full Accuracy	Reject Rate for Partial	Reject Rate for Full
40 / 0.84	94.05%	97.98%	63.95%	27.74%
Human	100.00%	100.00%	75.36%	33.58%

Table 2.3: Test accuracy for the COAD database for $N = 1$ using the EM algorithm.

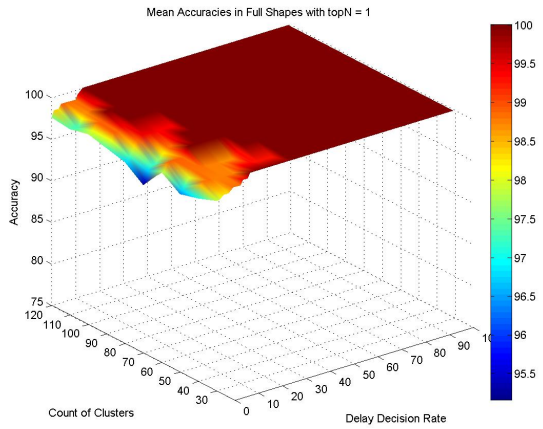
pared to human reject rate, but also somewhat lower accuracies compared to human accuracies), we next evaluated the semi-supervised CKMeans algorithm that was expected to do better with the fully drawn symbols, due to the imposed constraints. The results obtained using the CKMeans for clustering while keeping the other parts of the system unchanged, are given in the next sections.

2.3.4 Accuracy on the COAD database using CKMeans clustering

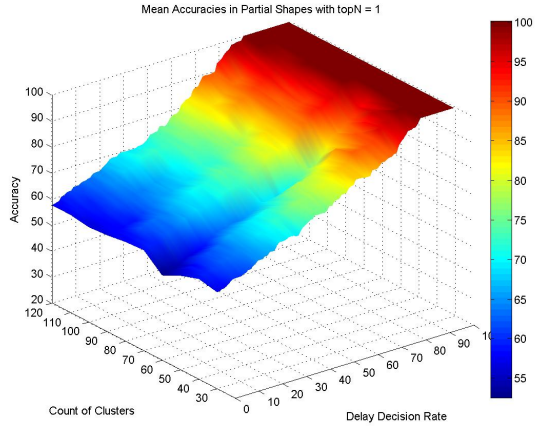
The performance surface of the system during validation with respect to the system parameters for the COAD database is shown in Figure 2-8, while representative points on this performance surface are listed in Table 2.4. The table is organized such that the first three rows present accuracies where the system is forced to make a decision ($C = 0$); while the last three rows present accuracies where the reject rates are close to human expert rates. Note that the accuracy result for full shapes at zero reject rate is comparable to recognition results without auto-completion, while the accuracies at human reject rates can be compared to human accuracies.

The best results are obtained with $K = 40, C = 0.74$ and $K = 40, C = 0.00$, depending on whether the system has the reject option or not, respectively. The corresponding test performance, obtained with these parameters, are shown in Table 2.5. One can see that the system achieves 100% accuracy for full symbols and 92.65% for partials, when the reject rates are even lower than the human reject rates. This counter intuitive result is explained in Section 3.4.1.

When $N = 2$ and $N = 3$, the accuracies increase since the system can make two/three guesses as to what the class of the object is. We again choose the best



(a) The performance surface for full symbols.



(b) The performance surface for partial symbols.

Figure 2-8: Validation performance for $N = 1$, on COAD database, using the CK-Means algorithm.

parameters in terms of validation set accuracy at close to human reject rates, which are found to be $K = 40, C = 0.88$ for $N = 2$ and $K = 40, C = 0.95$ for $N = 3$. At these settings, the test accuracies are given in Tables 2.6 and 2.7.

As mentioned before, the COAD database does not have explicitly defined training and test sets. So, in order to strengthen our results, we repeated the experiments using 5-fold cross validation where for each fold, 20% of the instances are separated for testing and the remaining instances are used for training. In each of these 5 experiments, the system parameters are optimized in a separate cross-validation as explained above, using only the training set allocated in that fold. For brevity, the

K/C	Partial Accuracy	Full Accuracy	Reject Rate for Partial	Reject Rate for Full
80 / 0.00	58.03%	96.77%	0.00%	0.00%
60 / 0.00	52.48%	97.85%	0.00%	0.00%
40 / 0.00*	58.49%	96.77%	0.00%	0.00%
80 / 0.83	90.46%	100.00%	75.00%	30.11%
60 / 0.79	88.71%	100.00%	75.76%	23.12%
40 / 0.74*	95.48%	99.31%	75.47%	24.19%

Table 2.4: Validation performance for $N = 1$ using the CKMeans algorithm on the COAD database. The rows with * indicate the parameters giving the best results.

K/C	Partial Accuracy	Full Accuracy	Reject Rate for Partial	Reject Rate for Full
40 / 0.00	54.94%	97.08%	0.00%	0.00%
40 / 0.74	92.65%	100.00%	70.82%	17.52%
Human	100.00%	100.00%	75.36%	33.58%

Table 2.5: Test performance for $N = 1$ using using the CKMeans algorithm on the COAD database.

results obtained with different tests sets are shown in Table 2.8 for only $N = 1$, along with the selected optimal parameter values. The row labeled *Exp 1* contains the results that are presented before. As illustrated in the table, the test results with different folds show low variance for the proposed classification method.

Discussion

For the COAD database, Tumen *et al.* [75] report a recognition accuracy around 96% for full symbols. This can be compared to the 97.08% accuracy obtained by our system during testing of full symbols, when reject was not an option (first row of

K/C	Partial Accuracy	Full Accuracy	Reject Rate for Partial	Reject Rate for Full
40 / 0.00	72.10%	99.27%	0.00%	0.00%
40 / 0.88	95.00%	100.00%	65.67%	18.25%
Human	100.00%	100.00%	61.74%	18.25%

Table 2.6: Test performance for $N = 2$ using using the CKMeans algorithm on the COAD database.

K/C	Partial Accuracy	Full Accuracy	Reject Rate for Partial	Reject Rate for Full
40 / 0.00	79.83%	99.27%	0.00%	0.00%
40 / 0.95	97.53%	100.00%	65.24%	17.52%
Human	100.00%	100.00%	55.07%	12.41%

Table 2.7: Test performance for $N = 3$ using using the CKMeans algorithm on the COAD database.

ID	K, C		Validation Accuracy	Validation Reject	Test Accuracy	Test Reject
Exp 1	40, 0.74	Full	99.31%	24.19%	100%	17.52%
		Partial	95.48%	75.47%	92.65%	70.82%
Fold 1	100, 0.78	Full	100.00%	28.19%	100.00%	24.63%
		Partial	87.69%	77.19%	90.00%	76.53%
Fold 2	80, 0.82	Full	100.00%	24.91%	100.00%	18.58%
		Partial	94.97%	73.83%	95.92%	68.70%
Fold 3	80, 0.82	Full	100.00%	25.51%	98.91%	22.34%
		Partial	91.62%	75.33%	88.00%	70.48%
Fold 4	60, 0.77	Full	100.00%	22.11%	100.00%	16.10%
		Partial	94.87%	71.24%	89.39%	69.59%
Fold 5	40, 0.75	Full	100.00%	25.05%	100.00%	20.53%
		Partial	94.19%	76.28%	98.28%	73.52%
Mean		Full	100.00%	25.15%	99.78%	20.44%
		Partial	92.67%	74.77%	92.32 %	71.76%

Table 2.8: Experiment results obtained using different test sets. Exp1 refers to the results given in Tables 2.4 and 2.5. The last row shows the mean of the accuracies and reject rates for the 5-folds.

Table 2.5). So, our system not only achieves better accuracy, but also does so while providing auto-completion.

More importantly, our system obtains 100% recognition accuracy in recognizing full symbols (second row, Table 2.5) at *lower* reject rates compared to humans. This may seem unintuitive at first, but it can be explained by two factors. First of all, human experts reject a full symbol F and tag it as ambiguous if it is a partial symbol of some other symbol S. However, if F has not occurred in the partial symbols of S in the training data, that information is exploited in the presented system. For instance, if the outer squares in Figures 2-6b to 2-6e are always drawn last, then Fig 2-6a is

not a partial symbol of any these symbols in practice. This information is captured by the system, as explained in Section 2.2.1.

Secondly, our system is biased towards performing better in full symbol recognition, when CKMeans is used with the constraints of not mixing full shape clusters. The algorithm is designed this way because, as mentioned earlier, an error in classifying a full symbol might cause more of a distraction to the user, than an error in classifying partial symbols.

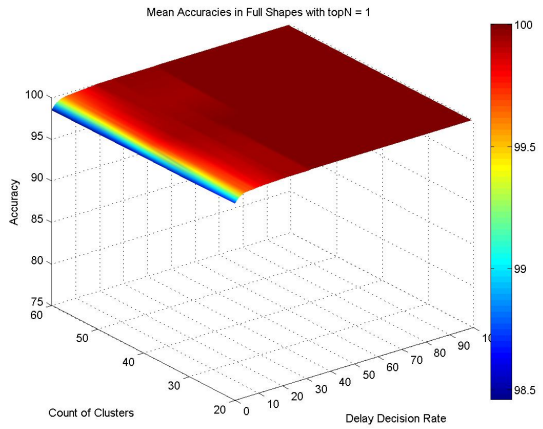
2.3.5 Accuracies on the NicIcon database using CKMeans clustering

The validation set performance of the system with respect to the system parameters for the NicIcon database is shown in Figure 2-9, while representative points on this performance surface are listed in Table 2.9. The first three rows of the table present accuracies on the performance surface at zero reject rate. Since no human expert labeling is done for this database, the last three rows in the table present the points at which less than 10% of the partials are rejected. The best results are obtained with $K = 20, C = 0.48$ and $K = 20, C = 0.00$, depending on whether the system has the reject option or not, respectively.

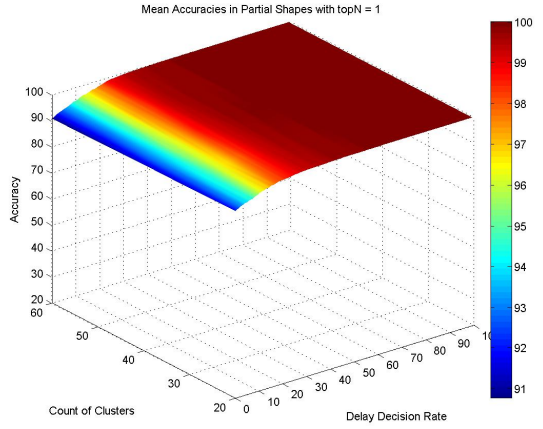
K/C	Partial Accuracy	Full Accuracy	Reject Rate for Partial	Reject Rate for Full
60 / 0.00	90.75%	98.43%	0.00%	0.00%
40 / 0.00	91.39%	98.52%	0.00%	0.00%
20 / 0.00 *	91.88%	98.64%	0.00%	0.00%
60 / 0.42	94.79%	99.10%	9.33%	1.80%
40 / 0.42	95.00%	99.37%	8.94%	1.96%
20 / 0.48*	95.92%	99.40%	9.74%	2.36%

Table 2.9: Validation performance for $N = 1$ using the CKMeans algorithm on the NicIcon database. The rows with * indicate the parameters giving the best results.

For $N = 2$ and $N = 3$, we do not change C and K. At these settings, the test accuracies are given in Tables 2.11 and 2.12.



(a) The performance surface for full symbols.



(b) The performance surface for partial symbols.

Figure 2-9: Validation performance surface for $N = 1$ using the CKMeans algorithm on the NicIcon database.

Discussion

As mentioned earlier, the NicIcon database is an easier database from the perspective of the auto-completion problem because the symbols in this database have more discriminative sub-symbols and fewer number of strokes. Even when the reject rate in partial symbols is 0%, partial symbol recognition accuracy is quite high (87.63%). As a comparison, the partial symbol recognition accuracy for the COAD database is only 54.94% as presented in Table 2.5.

In [82], the authors report a recognition accuracy of 99.2% for the NicIcon database. Our system achieves a recognition rate of 93.26% for full symbols, with 0% reject rate

K/C	Partial Accuracy	Full Accuracy	Reject Rate for Partial	Reject Rate for Full
20 / 0.00	87.63%	93.26%	0.00%	0.00%
20 / 0.48	93.06%	96.97%	14.33%	7.34%

Table 2.10: Test performance for $N = 1$ using using the CKMeans algorithm on the NicIcon database.

K/C	Partial Accuracy	Full Accuracy	Reject Rate for Partial	Reject Rate for Full
20 / 0.00	94.81%	95.71%	0.00%	0.00%
20 / 0.48	95.66%	96.51%	2.77%	1.80%

Table 2.11: Test performance for $N = 2$ using using the CKMeans algorithm on the NicIcon database.

as displayed in Table 2.10. Our recognition accuracy in full symbols is lower than the reported recognition rate on this database. However, our system is capable of performing auto-completion which is a valuable feature for sketch recognition applications.

The last experiment result in the NicIcon database for $N = 3$ is interesting. When $N = 3$, our system produces a higher recognition accuracy for partials than for fully completed symbols (97.47% vs. 96.75%). This result also supports the claim that auto-completion is well suited to the symbols in this database.

2.3.6 Comparison of clustering algorithms

In order to better observe the effect of semi-supervision on performance, we compared the accuracies obtained using each of the two clustering algorithms, for varying reject rates. In Figure 2-10, we present the comparison of EM and CKMeans in full symbol

K/C	Partial Accuracy	Full Accuracy	Reject Rate for Partial	Reject Rate for Full
20 / 0.00	97.47%	96.75%	0.00%	0.00%
20 / 0.48	97.57%	96.86%	0.30%	0.19%

Table 2.12: Test performance for $N = 3$ using using the CKMeans algorithm on the NicIcon database.

recognition using 80 clusters[1]. We can observe that the CKMeans performs better for all reject rates and achieves a high accuracy even for low reject rates.

Similarly, Figure 2-11 compares the partial symbol recognition accuracies, using the two clustering algorithms. We see that in the presence of semi-supervision, the accuracies increase when CKMeans is used not only for full symbols but also for partial symbols.

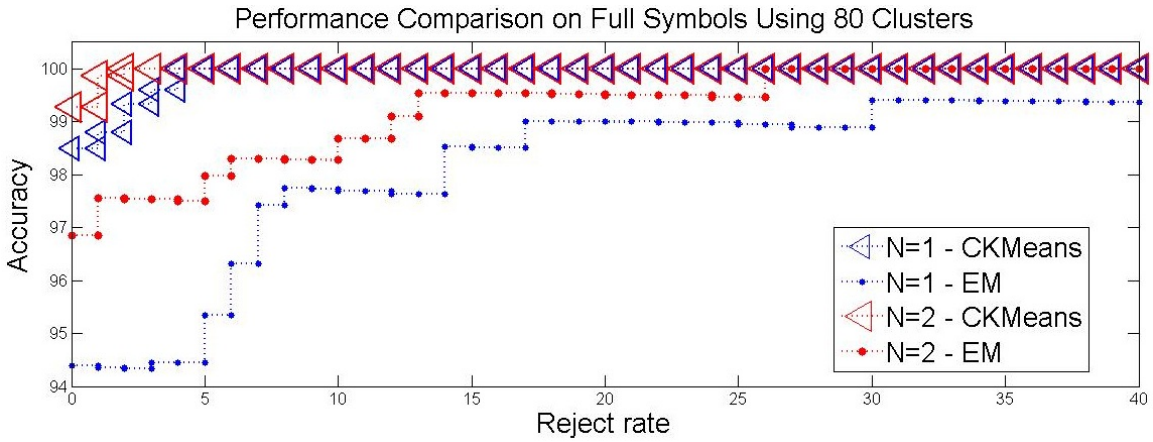


Figure 2-10: Comparison of full symbol accuracies on the COAD database, using EM and CKMeans with 80 clusters.

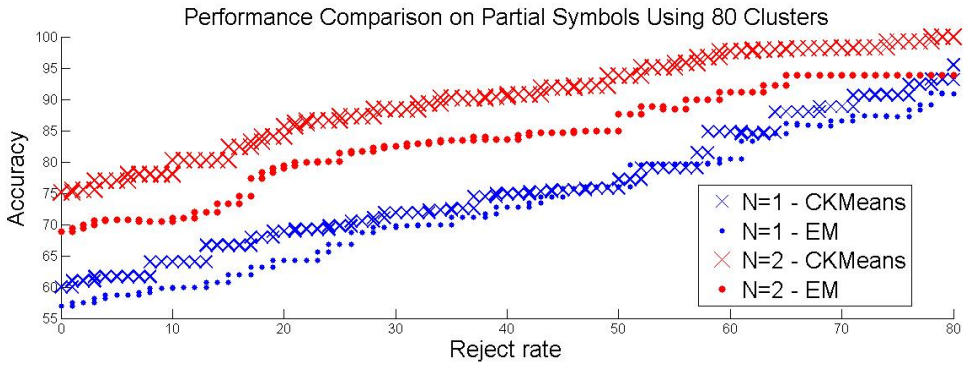


Figure 2-11: Comparison of partial symbol accuracies on the COAD database, using EM and CKMeans with 80 clusters.

[1]Similar patterns are observed for different cluster sizes as well.

2.3.7 Effect of supervised classification

As mentioned earlier in Section 2.2.4, we also conducted an experiment in order to observe the effect of supervised classification on system accuracy. We repeated the same experiments as above, using the NicIcon database, but removing the supervised classification component completely and using Eq. 2.3.

When the supervised learning step was eliminated, the best validation result at zero reject rate was obtained for 40 clusters. When the test performance was measured at this setting ($K = 40, C = 0.00$), we obtained the results given in Table 2.13. In this table, the first row shows the test performance on the NicIcon database using supervised classification (from Table 2.10) whereas the second row shows the test performance without the supervised classification. The contribution of the supervised classification is clear according to these results: Through the supervised classification step, the accuracy increases not only for full symbols, but also for partials.

Method	K/C	Partial Accuracy	Full Accuracy	Reject Rate for Partial	Reject Rate for Full
Proposed	20 / 0.00	87.63%	93.26%	0.00%	0.00%
No supervision	40 / 0.00	74.96%	89.37%	0.00%	0.00%

Table 2.13: The effect of removing the supervised classification step on the accuracies.

2.3.8 Implementation and runtime performance

We used LibSVM for the implementation of support vector machines [14], while the code for testing is written in Matlab. Classification of a single test instance takes roughly 0.07 seconds on a 2.16GHz laptop. So, the system runs in real time –as required for an auto-completion application.

2.4 Summary and discussions

We describe a system that uses semi-supervised clustering followed by supervised classification for building a sketch recognition system that provides auto-completion.

Our system approaches the auto-completion problem probabilistically and, although we have used a fixed confidence threshold during our tests, the confidence parameter can be modified by the user to specify the desired level of prediction/suggestion from the system. Experimental results show that predictions can be made for auto-completion purposes with high accuracies when the reject rates are close to that of a human expert. As described in the experiments, our system achieves 100.00% and 92.65% accuracies in the COAD database at human expert reject rates for full and partial symbols respectively. For the NicIcon database, 93.26% and 87.63% accuracies are obtained without rejecting any instances for full and partial symbols respectively. The system works in real time.

Few points are worth noting. First of all, there is a trade-off between accuracy and the ability to make predictions. For all values of N and K, increasing the confidence threshold improves accuracy, but it also increases the reject rate. It is important to locate points at which both reject rates and accuracies are acceptable. These points are found using a validation set, while in the actual application, the confidence threshold for a similarly selected K could be adjusted by the user.

Another point is that the system does not discriminate between full and partial symbols in its rejections. When the confidence threshold increases, more and more full symbol instances are rejected. However, what we would really like is to recognize full symbols well, at the cost of rejecting more partials if necessary. As we discussed in Section 2.3.6, integrating knowledge about the full symbols and using a semi-supervised clustering algorithm achieves this to some degree and also increases partial symbol recognition accuracy.

2.5 Future work

In this chapter, although we addressed on-line sketch recognition, we assumed that the scene contained only one object (either partial or full). In other words, we have not addressed issues that come up in the context of continuous sketch recognition where the scene may contain multiple objects. As shown in previous work [3], continuous

sketch recognition has its own challenges. In particular, the issue of how segmentation and auto-completion can be addressed simultaneously requires further research. It may be the case that an approach that is based on dynamic programming, may suffice and can be adapted to a scenario where the most recent object is partially drawn. It may also be the case that introducing the option for auto-completion may require modifications to the segmentation framework. More specifically, one has to make sure that the segmentation hypotheses generated by the recognition system allows only the latest object to be partial; all other groups computed by the segmentation step will have to correspond to fully completed objects.

Another question that arises naturally is how humans react to an interface which offers auto-completion. In order to figure out how and when auto-completion should be offered, user experiments need to be carried out. During those experiments, the parameters that we used, such as confidence threshold C and the number of choices to be offered, N, can be studied to find optimum parameter values.

Integrating machine learning methods for classifier combination into the system is also a future direction of research. During experiments, we observed that certain values of K do a better job at predicting full symbols, whereas others are better at predicting partials. Exploring a system that employs an ensemble of different K values can further boost the accuracies.

Chapter 3

Identifying visual attributes for object recognition from text and taxonomy

Attributes of objects such as *"square"*, *"metallic"* and *"red"* allow a way for humans to explain or discriminate object categories. These attributes also provide a useful intermediate representation for object recognition, including support for zero-shot learning from textual descriptions of object appearance. However, manual selection of relevant attributes among thousands of potential candidates is labor intensive. Hence, there is an increasing interest in mining attributes for object recognition. We introduce two novel techniques for nominating attributes and a method for assessing the suitability of candidate attributes for object recognition. The first technique for attribute nomination estimates attribute qualities based on their ability to discriminate objects at multiple levels of the taxonomy. The second technique leverages the linguistic concept of *distributional similarity* to further refine the estimated qualities. Attribute nomination is followed by our attribute assessment procedure, which assesses the quality of the candidate attributes based on their performance in object recognition. Our evaluations demonstrate that both taxonomy and distributional similarity serve as useful sources of information for attribute nomination, and our methods can effectively exploit them. We use the mined attributes in supervised and zero-shot learning settings

to show the utility of the selected attributes in object recognition. Our experimental results show that in the supervised case we can improve on a state of the art classifier while in the zero-shot scenario we make accurate predictions outperforming previous automated techniques.

3.1 Motivation

While much research in object recognition has focused on distinguishing categories, recent work has begun to focus on *attributes* that generalize across many categories [27, 24, 80, 45, 79, 42, 67, 7, 65, 23, 81, 12, 43, 58, 57]. Attributes such as *"pointy"* and *"legged"* are semantically meaningful, interpretable by humans, and serve as an intermediate layer between the top-level object categories and the low-level image features. Moreover, attributes are generalizable and allow a way to create compact representations for object categories. This enables a number of useful new capabilities: *zero-shot learning* where unseen categories are recognized [80, 45, 67], generation of textual descriptions and part localization [24, 79, 23], prediction of color or texture types [27], and improving performance of fine-grained recognition tasks (*e.g.*, butterfly and bird species or face recognition) [80, 42, 12, 43, 57] where categories are closely related.

However, using attributes for object recognition requires answering a number of challenging technical questions — most crucially, specifying the set of attributes and the category-attribute associations. Most prior work uses a predefined list of attributes specified either by domain experts [80, 45, 12] or researchers [24, 42, 81, 43], but such lists may be time-consuming to generate for a new task, and the attributes in the generated list may not correspond to the optimal set of attributes for the task at hand. A natural alternative is to identify attributes automatically, for example, from textual descriptions of categories. However, this is challenging because the number of potential attributes is large, and evaluating the quality of each potential attribute is expensive.

We present a system that automatically discovers a list of attributes for object

recognition. As we approach the problem from a computer vision perspective, we are mainly concerned with "visual" attributes that directly relate to the appearance of objects, such as "*red*" or "*metallic*". However, an attribute that relates with visual qualities in general may not be selected by our system if it does not help the recognition task, *e.g.*, "*metallic*" is not a useful attribute if the recognition task is to classify car brands. In contrast, the word "*fragrant*" does not refer to a visual quality, however due to its indirect correlation to visual features (*e.g.*, its link to flowers), it may be selected as a useful attribute for object recognition by the proposed method. In the remainder of this chapter we use the term "visual attribute" to refer to any word that may help object recognition from images.

Our main contributions are as follows. Firstly, we introduce two methods to select words in a text corpus that are likely to refer to visual attributes. One of the methods we propose uses a taxonomy defined over categories and promotes words whose occurrence in textual descriptions of categories is coherent with the given taxonomy. The other method builds upon the previous one and integrates distributional similarity of words into the attribute selection process. Secondly, we propose a way to assess the quality of a candidate word as an attribute for object recognition from images. In the experiments, we provide evaluations of the proposed attribute selection strategies for effectively identifying attributes, in the plant and animal domains, and present the efficacy of the proposed techniques at selecting visual attributes. Furthermore, we analyze the mined attributes semantically and then use them for plant and animal identification tasks.

We use three input sources in the proposed methods: textual descriptions and image samples of categories and a taxonomic organization of the object domain. In the plant identification task, the goal is to identify a plant species from the visual appearance of its foliage. We use the plant foliage image dataset provided in ImageCLEF'2012[1] plant identification task [29]. This dataset contains $9,356$ foliage images of 122 species of which $6,689$ are for training and $2,667$ are for testing. For this dataset, we mine a set of text documents containing encyclopedic information on cat-

[1] http://www.imageclef.org/2012

egories from the web, using Wikipedia[2], Encyclopedia of Life[3] and the Uconn Plant Database[4]. For the animal identification task, we use the animals-with-attributes (AwA) database provided by [45] to evaluate our approach. This is a popular database to test attribute-based recognition and zero-shot learning approaches. This dataset contains 30,475 images of 50 animals where 24,295 images of the selected 40 animals are used for training and 6,180 images of the remaining 10 classes are reserved for testing in the zero-shot learning setting. Similar to the plant identification, we mine textual descriptions for each of the 50 animals in that set using Wikipedia and A-Z Animals.[5] In both of the recognition tasks, the challenge is to find the words referring to visual attributes in the mined documents. We test the effectiveness of the automatically selected attributes for recognition in both zero-shot learning and in traditional supervised learning settings. [6]

Our approach consists of two main components. After reviewing related work in 3.2, we describe a method for assessing the visual quality of a proposed attribute for object recognition in Section 3.3. The assessment procedure involves training a binary attribute classifier, where the quality of a candidate word depends on the success of the attribute classifier. Classification-based attribute selection is effective but computationally expensive; consequently, in Section 3.4, we propose a set of techniques for *nominating* candidate attributes that are likely to be of high visual quality: we leverage multi-level discriminability across a category taxonomy, and distributional similarity of the words in the text corpus. Our nomination process takes feedback from the visual quality assessment of candidate attributes, making increasingly accurate predictions as it learns more about the types of words that are found to be of high quality. Once the set of attributes is determined, in Section 3.5, we illustrate how the selected attributes can be used for object recognition in two different settings. Finally, in the experiments section (Section 3.6) we present experiments to

[2] http://en.wikipedia.org/wiki/Main_Page
[3] http://eol.org/
[4] http://www.hort.uconn.edu/plants/
[5] http://a-z-animals.com/
[6] All the collected textual descriptions are available online at: https://drive.google.com/file/d/0Bx-64dmWqUHIT09JRGZDOGxPNkk/view?usp=sharing

compare attribute selection strategies. Then, the selected attributes are used for classification of categories in two challenging recognition tasks.

3.2 Related work

Although most of the literature on attribute-centric recognition focuses on working with a predetermined list of attributes, a few alternatives propose methods to select attributes interactively [58, 57] or automatically [7, 65]. The interactive methods first identify local image patches that are important for recognition and then use human supervision to check whether or not these patches refer to attributes. Unlike these methods, we would like to take advantage of a text corpus and select attributes automatically. Berg *et al.* [7] also use a text corpus, but instead of learning which attributes are valuable for recognition from text using textual features, they iteratively test the most frequent words in the corpus to find attributes. We show that an intelligently guided search identifies effective attributes much more rapidly.

There is also previous work in the literature to identify words referring to visual characteristics [5, 11, 21]. In [5] Barnard and Yanai fit a Gaussian mixture model to image regions and determine the "visualness" of a word based on the entropy of the distribution. Boiy *et al.* [11] mine words having visual information using corpus-based association techniques where words appearing more in the texts of visual corpus rather than non-visual corpus are selected. In [21] authors use several strategies to mine visual text from large text corpora. First, they generate a graph between adjectives based on distributional similarity and apply bootstrapping to select visual nouns and adjectives. Second they construct a bipartite graph between visually descriptive words and their arguments and use label propagation to extend the list of visual words. Finally, they integrate visual features to improve performance. In comparison, we propose methods that utilize a taxonomy over categories and distributional similarity of words to automatically discover attributes of categories that are likely to refer to visual characteristics.

Another related work [64] involves finding discriminative codes for individual im-

ages rather than for categories. In their work Rastegari *et al.* create a system to encode each image with a binary code to balance discrimination between categories and learnability of the individual attribute classifiers. Although there is no direct semantic mapping of the discovered codes, they achieve state-of-the-art recognition results on Caltech256 [30] and ImageNet [18] databases. In contrast to their work, we define attributes on the category level and rely on a text corpus to automatically mine semantically meaningful and discriminative attributes.

In [86], Yu *et al.* design a category-attribute matrix on the *known* categories where they balance the separation of the categories while also considering the learnability of the attribute classifiers. In order to perform zero-shot learning, they use human supervision to create a similarity matrix between the novel and the known categories while using the trained attribute classifiers. While we also design a category-attribute matrix, we use no human-supervision in the process other than supplying the readily-available taxonomy on the categories. Moreover, since we use a text corpus to mine the attributes, the attributes we discover can be directly mapped to semantically meaningful units.

The most similar prior work to ours is [65], where Rohrbach *et al.* use state-of-the-art natural language processing techniques and provide experiments for several linguistic knowledge bases for mining attributes. However, there are key differences. Rohrbach *et al.* consider PART-OF relations encoded in WordNet [52]. In contrast, we mine attributes using a taxonomy defined on the categories and consider the whole text so our method can discover attributes referring to color or context that cannot be explained with PART-OF relations. Furthermore, expanding the set of candidate attributes to all words requires accurate nomination heuristics; in our approach, we provide this by leveraging the object category taxonomy and distributional similarity.

Object classification using a taxonomy has also been studied before. In [31] Griffin and Perona describe a way to automatically learn hierarchical relationships between images of categories and use this taxonomy in the recognition task. Deng *et al.* [19] show that there is a correlation between the structure of the semantic hierarchy of the WordNet and visual confusion between the categories. They present a cost function

based on the WordNet hierarchy for classification of 10000 categories and show that it produces more semantically meaningful classification results. By defining a taxonomy over categories, Binder *et al.* [8] train an ensemble of local SVMs on various levels of the taxonomy and use trained classifiers in the recognition task. In contrast to previous lines of work that utilize a taxonomy to improve speed and recognition accuracy from images, we rely on the readily available taxonomy of life to discover attributes of categories in a text corpus that help object recognition.

Finally, in [26], a unimodal topic model that integrates textual and image features is built for the tasks of computing word association and similarity. More recently, in [71] the authors combine visual attribute classifiers with text-based distributional models for finding word associations. In both of these papers, improved results are obtained when textual and image features are used together. However, these lines of work aim to ground natural language semantics in visual features, rather than using language to improve object recognition from images.

3.3 Assessing the visual quality of attributes

In the textual description of a category such as a plant or animal, only a very small fraction of words refer to attributes — such as *"legged"*, *"green"*, *"big"*, *"ugly"* or *"sharp"*. Moreover, some of these words refer to very high level qualities (*e.g.*, *"ugly"*) that may not be robustly detectable using automatic methods. Also, attributes that are beneficial might change depending on the recognition task. For example, the word *"green"* can be used as an attribute for various tasks, but it is not a useful attribute for plant identification using images of foliage, as most plant foliage is green.

In this section, we present a method to test whether a given candidate word refers to an attribute that can be recognized using visual features. The method is based on the assumption that a word denoting an attribute of an object category will appear frequently in its description. Furthermore, we require from an attribute classifier trained with a proportion of object categories having the attribute versus others, to do a good job in separating *novel* categories having the attribute from

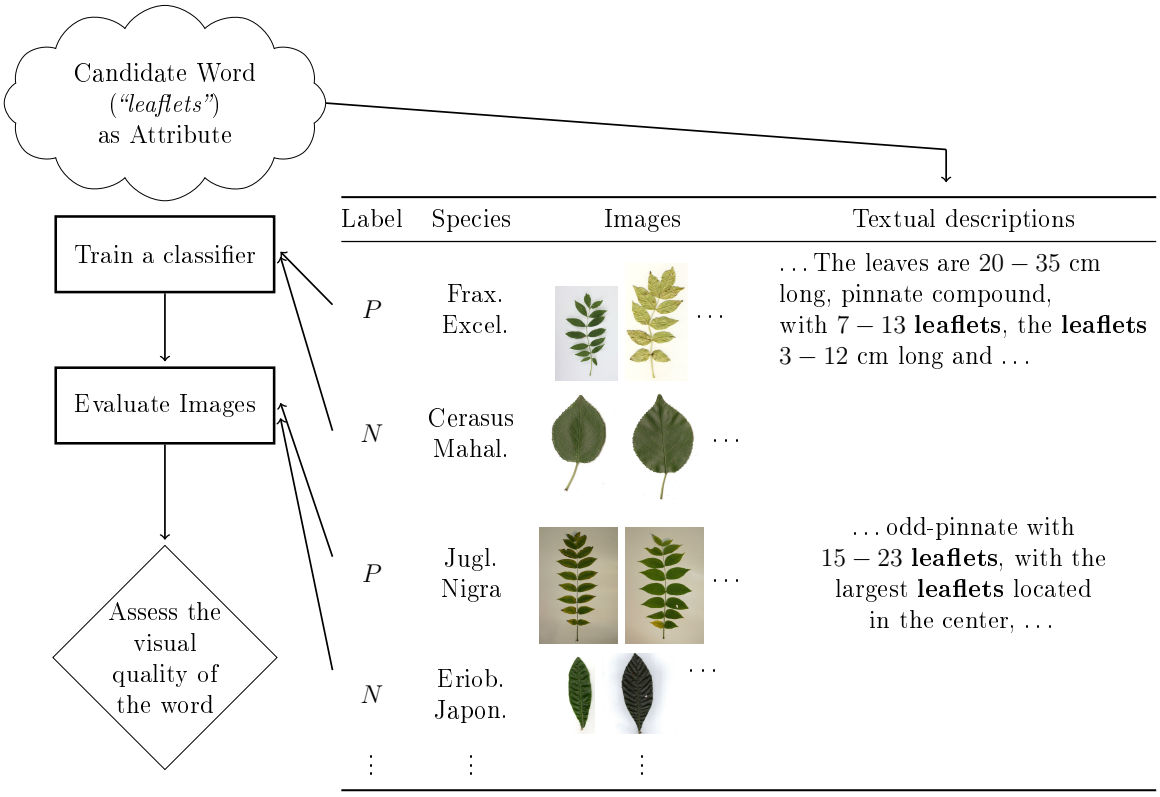


Figure 3-1: The flow for how the visual quality of a candidate word is assessed. Each category has a set of images and textual descriptions. Given a candidate word, each category is associated with a positive (P) or negative (N) label for this candidate, using its textual descriptions (This is unlike previous works [64, 20, 7] where instances are associated with labels). Images of half of the P and N categories are used to train an attribute classifier and the classifier is evaluated on the remaining images. The candidate word is assessed based on the classifier responses on the evaluation images.

others. Figure 3-1 summarizes our process for assessing whether a candidate word is accepted as an attribute; each step is described in detail in the following subsections.

3.3.1 Constructing the training set

Given a candidate word as an attribute, all object categories in the image collection are *automatically* labeled as either having or not having the attribute, based on their textual descriptions. The categories having the attribute are associated with the label positive (P) while the remaining categories are associated with the negative (N) label. This is unlike previous works [64, 20, 7] that use instance-attribute rather than category-attribute association.

The category-attribute association is based on the occurrence frequency of the candidate word in the description of that category in the text corpus. Specifically, we compute the mean and standard deviation of the word frequency across all categories, and then associate a category with P if the frequency of the word in descriptions of that category is at least one standard deviation above the mean frequency. All other categories are associated with the label N. We have tried various other methods for finding the category-attribute associations but saw that this method works quite well in practice.

3.3.2 Training the attribute classifier

The P and N categories identified in Section 3.3.1 are split into training and evaluation sets, using a 50/50 split. If category is placed in the training set, then all image instances for that category are used for training, and vice versa for the evaluation set. In this way, we avoid attributes specific to a single category that cannot be shared across categories. To be selected, an attribute must be recognizable in novel categories that are unseen in the training data.

Using the P and N image instances reserved for training, we train a binary attribute classifier that learns to differentiate between them. In other words, given a candidate attribute (*e.g.*, *"striped"*), the classifier is trained to separate the set of images labeled as having this attribute (zebras, tiger, ...) or not (lion, panther, ...) based on their textual description. The trained attribute classifier is used to detect the attribute a in a given image x, with its output interpreted as $p(a|x)$.

Our experiments focus on plant and animal identification tasks. In the plant identification experiments, we use a binary attribute classifier that operates on a set of features extracted from images; specifically histogram of gradients [16], shape context [6] and local curvature on the object boundary. Each attribute classifier is trained using these feature descriptors of the images in the P and N sets reserved for training.

For the experiments we perform on the Animals with Attributes (AwA) dataset, we train the attribute classifiers using the pre-computed descriptors (color, local self

similarity, oriented gradients, rgSIFT, SIFT and SURF histograms) supplied by the authors of the AwA database [45] as well an additional feature descriptor extracted using a convolutional neural network. The feature descriptors of 40 training categories specified in [45] are used in training of the attribute classifiers. During assessment of a candidate word, we train a binary linear SVM as the attribute classifier for each feature descriptor.

3.3.3 Assessing the visual quality

The trained attribute classifier is used to produce $p(a|x)$ for each image instance in the evaluation set. We then analyze the probability distributions obtained by the positive and negative samples in the evaluation set, to determine whether the candidate word is accepted as an attribute. The candidate word is accepted as an attribute if the distribution for the instances of the P categories is significantly greater than the distribution of the instances of the N categories. In order to compare the distributions we perform a t-test at $p < .01$. While precision of the attribute classifier at separating positive and negative instances has been used to assess visual quality of a candidate before [20, 7], we preferred to use a t-test which allows us to a detect statistical difference between the distributions of positive and negative instances.

Figure 3-2 presents the histograms of the probability distributions of the positive and negative evaluation instances for two words that are candidate attributes: *"deciduous"* and *"leaflets"*. The candidate word *"deciduous"* fails the assessment, as the classifier assigns essentially the same distributions to the instances of P and N categories in the evaluation set. In contrast, the distribution for the instances of the P categories for the word *"leaflets"* is skewed to the right compared to that of the instances of N categories. We take this to mean that the word *"leaflets"* corresponds to some visual characteristics.

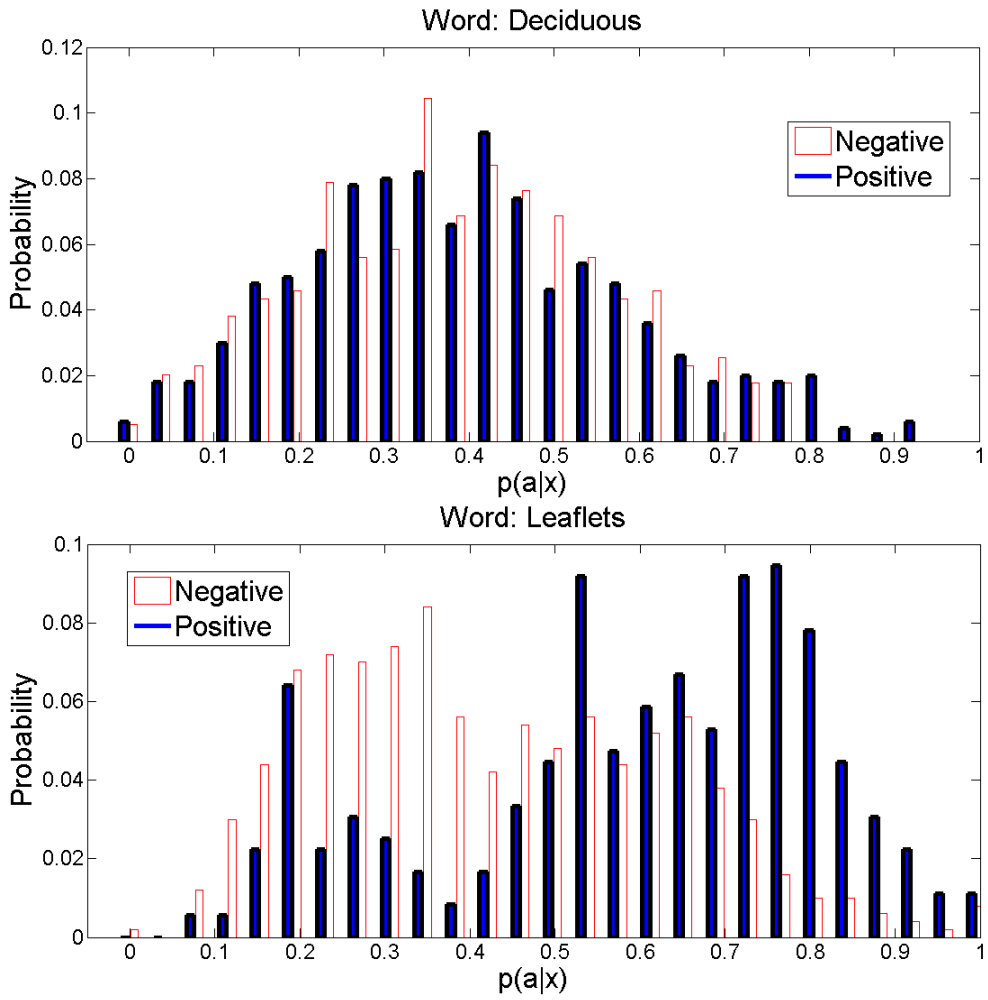


Figure 3-2: Two sample words (*"deciduous"* and *"leaflets"*) are assessed for visual quality and the histograms for the attribute classifier predictions are presented. *"Deciduous"* is not accepted because the the classifier predictions are similar for the instances having (positives) and without (negatives) the attribute. On the contrary, *"leaflets"* is accepted because the attribute classifier produces higher probabilities for the instances having the attribute.

3.4 Attribute candidate ranking

The previous section describes an effective procedure for determining if a word is an attribute, but it requires training a binary classification system. Doing so for several thousand candidate attributes can be very time consuming. Hence, we propose techniques to rank the candidates so that the most promising ones are considered first, allowing us to obtain a good set of attributes without iterating through all possible

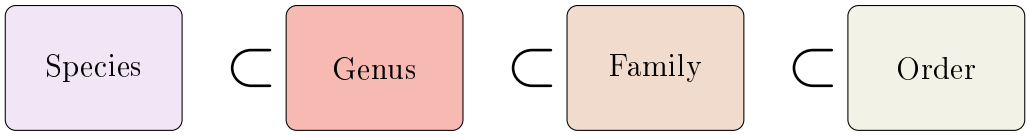


Figure 3-3: The lowest four ranks in the hierarchy of biological classification. Each species is also a member of a genus, family, order, *etc*.

candidates.

The first technique we propose measures each word's effectiveness as an attribute based on its ability to discriminate not only individual categories, but also higher-order sets of categories as defined by a taxonomy on categories. For example, a good attribute should distinguish not just cars and trucks, but higher-order categories, like vehicles.

The second method we propose organizes all candidate words into a hierarchy, using *distributional similarity*, so that words with similar distributional properties are in similar parts of the hierarchy. As we assess candidate words, we obtain firm evidence about the visual quality of individual words. This information is then propagated to the neighbors of the assessed candidate word in the hierarchy: *i.e.* if a word is found to be a good attribute, then distributionally-similar words will be ranked higher as well, and vice versa. We now discuss these ideas in detail.

3.4.1 Use of object taxonomy

The living organisms have a taxonomy where organisms are categorized based on similarities or common characteristics. The lowest four ranks of this taxonomy are displayed in Figure 3-3: each species is associated with a genus, family, and order. Two species in the same genus are therefore more similar to each other than to other species in different genera. Likewise, two species in the same family are more similar to each other compared to other species in different families.

In the thesis we are working on classification of plants and animals from images and their biological taxonomy is readily available. [7] Assuming this conceptual taxonomy

[7] The taxonomies of plants and animals we use are available online at: https://drive.google.com/file/d/0Bx-64dmWqUHISHhjbWYwOXUxZFk/view?usp=sharing

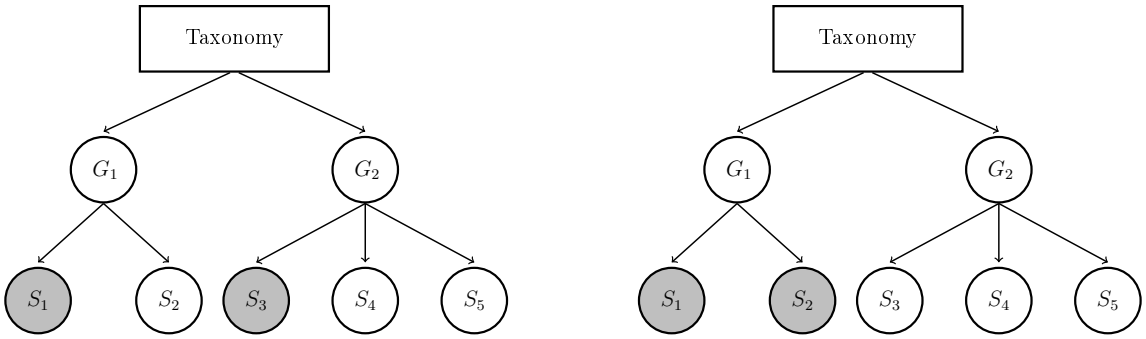


Figure 3-4: A toy example where a taxonomy over 5 distinct species and 2 genera is illustrated. We present the division of the species as positive and negative for two candidate words where the nodes that are gray are positive and the others are negative. On the given taxonomy, picking the word on the left splits the species that are in the same genus while the word on the right respects the taxonomy.

has some correspondence in the visual appearance of each object category, we would like to choose attributes that match it. Such attributes should help us discriminate between highly disparate object classes.

We motivate the use of taxonomy in finding words that are likely to denote visual characteristics with a toy example. Suppose we have a taxonomy defined over 5 species (the leaf nodes) and 2 genera (the middle nodes), as shown in Figure 3-4 and we have a dictionary of 2 words. We would like to pick one of the words as a candidate attribute and the task is deciding which one to pick. As described in Section 3.3 each category (species) is associated with either a positive (gray nodes) or negative (white nodes) label depending on the occurrence frequency of the word in textual descriptions of the category. Now, consider the first word that creates the labeling on the left. This word creates a positive set using categories S_1 and S_3, but these categories belong to different genera. In contrast, the second word that generates the tree on the right side of the figure, illustrates a candidate that respects the taxonomic organization of categories. We hypothesize that words that conform to the taxonomy, such as the second word, will be more likely to have meaningful visual properties, and the results in Section 3.6 bear this intuition out. We now describe a ranking procedure that will favor such words.

Formally, suppose we are given a set of categories $S = \{S_1, S_2, \ldots, S_M\}$, where

each category is represented by a set of text documents $S_i = \left\{t_1^i, t_2^i, \ldots, t_{N_i}^i\right\}$. Assume further that each document has a vector space representation based on tf-idf (term frequency inverse document frequency [50]), where the length of the representation is the same as the dictionary size. Finally, denote by d_{ij} a parametric distance between a pair of text documents t_i and t_j. We will parametrize the distance using a weight vector on the words; words whose discrimination pattern is consistent with the category taxonomy will get higher weights. We pose a constrained optimization problem for this purpose.

For concreteness, we focus on the recognition tasks where the relevant groups are species (S), Genera (G), and Families (F). We pose the following constraints:

$$\begin{aligned} d_{ij} + 1 &\leq d_{kj} \qquad \forall t_i, t_j \in S_I \text{ and } \forall t_k \notin S_I \\ d_{ij} + 1 &\leq d_{kj} \qquad \forall t_i, t_j \in G_I \text{ and } \forall t_k \notin G_I \\ d_{ij} + 1 &\leq d_{kj} \qquad \forall t_i, t_j \in F_I \text{ and } \forall t_k \notin F_I \end{aligned} \tag{3.1}$$

The first constraint states that two documents of the same species should be closer to each other than to documents of the remaining species. Similarly, the second constraint states that two documents belonging to the species of the same genus should be closer to each other compared to the documents of the remaining genera. The third constraint enforces the same condition at the level of families.

Now suppose $t_i \in S_I$, we define:

$$\begin{aligned} \vec{\delta}_{ij} &= |f_{\text{tf-idf}}(t_i) - f_{\text{tf-idf}}(t_j)| \\ d_{ij} &= \left\langle \vec{w}_I, \vec{\delta}_{ij} \right\rangle \end{aligned} \tag{3.2}$$

where $f_{\text{tf-idf}}$ is a function computing the vector of tf-idf values of the dictionary words for its input; $\vec{\delta}_{ij}$ is the absolute difference vector between the tf-idf vectors of t_i and t_j, $\vec{w}_I$ is the weight vector for the object category S_I and $\vec{w}_I$ is the weight vector for the object category S_I and $\langle , \rangle$ denotes dot product.

Our procedure for learning the weights is inspired by the metric learning approach

of Frome *et al.* [28]. Suppose $t_i \in S_I$, $t_k \in S_K$; denote by $\vec{w}_{IK}$ the concatenation of the weight vectors for S_I and S_K; let $\vec{x}_{ijk} = [-\vec{\delta}_{ij}\vec{\delta}_{kj}]$, the concatenation of $\vec{\delta}_{ij}$ negated and $\vec{\delta}_{kj}$. Then, the first constraint in Equation 3.1 can be rewritten as $\langle \vec{w}_{IK}, \vec{x}_{ijk} \rangle \geq 1$. We denote the next two constraints that are defined on the genus and family levels similarly, and we define a loss function over all constraints and all triplets:

$$\sum_{Constraints} \sum_{ijk} \lfloor 1 - \langle \vec{w}_{IK}, \vec{x}_{ijk} \rangle \rfloor_+ \tag{3.3}$$

where $\lfloor z \rfloor_+$ denotes the thresholding function $max(0, z)$. After adding a regularization penalty, the objective function becomes:

$$\frac{1}{2} \|\vec{w}\|^2 + C \sum_{Constraints} \sum_{ijk} \lfloor 1 - \langle \vec{w}_{IK}, \vec{x}_{ijk} \rangle \rfloor_+ \tag{3.4}$$

where $\vec{w}$ is the concatenation of the weight vectors of all categories and C is the regularization parameter. We select C using cross validation, using the value that minimizes the loss function on held-out data.

Once the regularization parameter is selected, the optimization problem can be solved using a row-action method similar to [28]. The weight vector of each species is updated by iterating over all constraints, and the triplets associated with them as follows:

$$\begin{aligned} \alpha_{ijk} &\leftarrow \left\lfloor \frac{1 - \langle \vec{w}_{IK}, \vec{x}_{ijk} \rangle}{\|\vec{x}_{ijk}\|^2} + \alpha_{ijk} \right\rfloor_{[0,C]} \\ \vec{w}_I &\leftarrow \vec{w}_I - \sum_{ijk} \alpha_{ijk} \vec{\delta}_{ij} \\ \vec{w}_K &\leftarrow \vec{w}_K + \sum_{ijk} \alpha_{ijk} \vec{\delta}_{kj} \end{aligned} \tag{3.5}$$

where $\lfloor z \rfloor_{[0,C]}$ denote the function $min(C, max(z, 0))$. We continue iterating until the change in weights falls below a threshold. Unlike [28], where the authors define constraints on the instances of categories, we have constraints defined over the taxonomy of categories and we do not enforce non-negative weights since words with negative

weights can also indicate potential attributes.

Since solving this constrained optimization problem relies on generation of document triplets, let us elaborate on the number of triplets that will be generated. Consider our first constraint in Equation 3.1, denote the number of categories by M, and the number of documents per category by N. Then the number of triplets that will be generated is $O(M^2N^3)$. Working with that many triplets might be infeasible, so when generating triplets we select only a subset of all possible triplets. While forming triplets of the form $\langle ijk \rangle$, we select a document k only if it is in the R nearest neighbors of i based on its tf-idf representation, reducing the number of triplets to $O(MN^2R)$. We set $R = 50$ during all experiments. In the experiments, we consider the most frequent $2,000$ non-stoplist words in the text corpus to create tf-idf representations, thus the weight vector of each category is of length $2,000$.

Once the weight vectors for all categories are learned, we assign a weight for each word in our vocabulary by computing the mean absolute weight of the word over the categories. Words that are shared among categories and obey the category taxonomy will get higher weights, and therefore will be considered first as potential attributes.

We have implemented the described optimization using C#. [8] It takes 31 and 276 seconds for the optimization to be completed on the AwA and the ImageClef datasets respectively using a computer having Intel i7 2.00 GHz processor.

3.4.2 Integrating distributional similarity

While taxonomic discriminability is a powerful feature for predicting whether a word will be a useful attribute in general, it ignores the word's meaning and considers only co-occurrence with category-labeled documents. We hypothesize that if a word has a high value as an attribute, then words with similar meanings should also have high value, and vice versa. Word meanings — known as *lexical semantics* in linguistics — can be difficult to pin down. This is particularly true in technical domains such as plant/animal biology, where annotated resources such as WordNet may have low

[8] Available online at: https://drive.google.com/file/d/0Bx-64dmWqUHIVzJ3djlCUDQxRjQ/view?usp=sharing

coverage. We follow an alternative, data-driven approach to lexical semantics, motivated by the *distributional hypothesis*, which asserts that words with similar meanings tend to appear in similar linguistic contexts [35]. This bears directly on our problem of identifying words that are attributes. For example, if the word *"lobed"* is found to be a visual attribute, then words that appear in similar contexts to *"lobed"* (*e.g.*, , *"serrated"*, *"oblong"*) are also likely to be attributes and should be prioritized for testing. Conversely, if a word such as *"Western"* is found not to be a visual attribute, then words that appear in similar contexts to *"Western"* (*e.g.*, , *"Eastern"*, *"Chinese"*) are unlikely to be attributes (despite having high taxonomic discriminability), and can be tested later.

We use a hierarchical clustering of words to capture word similarity. Each word is represented by a vector of frequencies, where the vector of a word represents its co-occurrence with neighboring words [53, 9]. In order to construct the co-occurrence vector of a word we use a context window of five words on either side of the target word where the vector dimensions are constituted by the most frequent 2000 non-stoplist words in the text corpus. Finally, we apply a graph clustering algorithm [68] to the word representations, obtaining a word dendrogram (Figure 3-5) [9].

To see how word similarity can help, consider the toy example shown in Figure 3-5. Weights for each word are estimated to be $w_1 = w_2 = 1, w_3 = w_4 = w_5 = 0.8$, using the procedure described in Section 3.4.1. Initially, we pick W_1, which is tied for the highest weight. However, suppose that W_1 fails the visual quality assessment. The word W_2 is tied for the next highest weight, but it is distributionally similar to W_1. Since W_1 is not selected as an attribute, we downweight our prediction for W_2, and try another word instead. Note that this idea applies to any hierarchical clustering of words, so we could use other word features instead of co-occurrence-based features to cluster the words and use the same idea.

The key advantage of the word dendrogram is twofold. Firstly, it allows us to integrate distributional similarity into the candidate selection process. Secondly, by

[9] The word dendrograms for the Awa and the ImageClef datasets are available online at: https://drive.google.com/file/d/0Bx-64dmWqUHIcTN2eEthQnBSMDA/view?usp=sharing

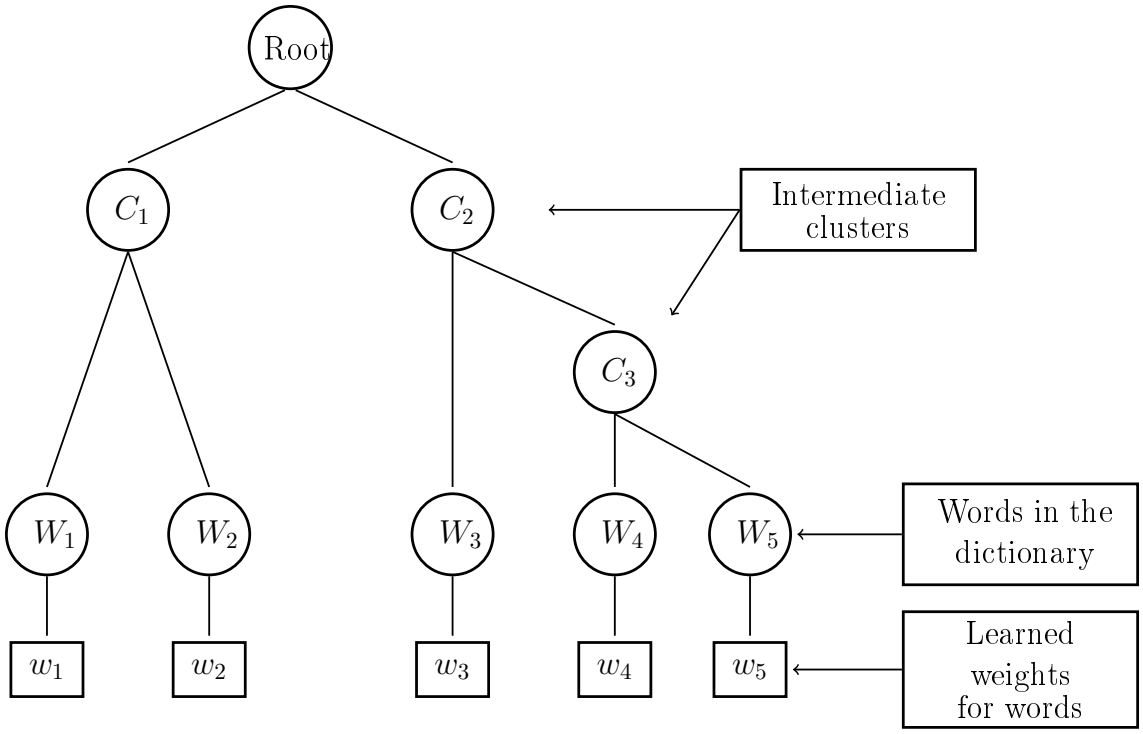


Figure 3-5: The graphical model based on the hierarchical clustering of 5 words. The learned weights for words are used to compute likelihoods of nodes being effective visual words and each edge between nodes is governed by a compatibility function favoring the same visual effectiveness for neighbors.

propagating information about visual quality assessment between related words, better candidates can be selected as the candidate selection progresses. Initially, the dendrogram helps to smooth the learned weights using taxonomy computed in Section 3.4.1. As we assess candidate words (Section 3.3), these initial estimates are replaced with hard evidence. By propagating this evidence through the word dendrogram, we can avoid wasting effort assessing unpromising words whose near neighbors have already failed assessment.

We operationalize this idea in the framework of belief propagation [62, 85], treating the word dendrogram as a large graphical model containing binary random variables x_e for both words and word clusters. We treat visual quality assessment result as a latent variable x_e for node e, and perform inference on the distribution $P(\vec{x}_{1:E}|\vec{w}_{1:E})$, where $\vec{w}_{1:E}$ is the local evidence for all nodes $1:E$. This probability is proportional to $P(\vec{w}_{1:E}|\vec{x}_{1:E})P(\vec{x}_{1:E})$, where the first term indicates the likelihood and the second term

Algorithm 1 Procedure for nominating and assessing attributes.

function PROPOSE ATTRIBUTES(W)
    Estimate taxonomy-based discriminability for all words (Section 3.4.1)
    Build a dendrogram based on distributional similarity (Section 3.4.2)
    while there are untested words **do**
        Apply belief propagation to estimate $P(x_e|\vec{w})$ for all words
        Assess the visual quality of the top-scoring untested word e (Section 3.3), and clamp x_e
    end while
end function

indicates the prior. After normalizing the learned weights for words between 0 and 1, the likelihood for each node is set equal to the normalized weight of the respective word. The prior is governed by a compatibility function, and in our experiments for neighboring nodes $x_e^{'}, x_e$ we define:

$$P(x_e^{'}|x_e) = \frac{1}{2}\begin{bmatrix} \alpha & (1-\alpha) \\ (1-\alpha) & \alpha \end{bmatrix}. \tag{3.6}$$

where α is a constant greater than 0.5. We experimented with various values of α and set it 0.6 in all experiments, obtaining the best speed at selecting visual attributes. For words whose visual quality has been assessed, we can *clamp* x_e to the true value. By applying belief propagation, information from these clamped nodes is propagated to neighboring nodes, with diminishing influence as we move across the dendrogram.

We use Algorithm 1 for nominating and assessing candidate words. With this strategy, we adaptively search the set of possible attributes, focusing our initial efforts on the most promising words while also taking into account the assessed words during later iterations.

3.5 Attribute-based classification

After generating a list of attributes as described so far, the discovered attributes can be used for object classification. In order to use the selected attributes in object recognition, we train attribute classifiers (one per attribute), such that each classifier

is trained to discriminate between the images of positive and negative classes (see Section 3.3.1) corresponding to that attribute. We use the attribute classifiers for supervised attribute-based classification of plants and zero-shot learning of animals. During the direct similarity-based classification experiments for zero-shot learning, we utilize the learned weight vectors of categories in Section 3.4.1. Below, we discuss these approaches.

3.5.1 Supervised attribute-based classification

Using attributes we apply the traditional supervised learning paradigm to recognize test images. In order to extract training features, we apply the attribute classifiers on all the training images of each category. For each image we create a feature vector that is the same length as the the number of attributes containing attribute classifier responses. Next, an SVM classifier is trained [14] using the extracted features. During testing, we extract the feature vector from a test instance (image) by applying the attribute classifiers and concatenating the classifier responses. The extracted feature vector is then classified by the trained SVM classifier. Supervised attribute-based classification offers advantages over traditional classifiers, since the feature vector is compact and the underlying representation is interpretable to humans.

3.5.2 Zero-shot learning

In zero-shot learning, the aim is to learn to recognize novel categories using only their textual descriptions. For instance, having trained attribute classifiers *has-a-torso* and *has-a-tail*, zero-shot learning enables us to label an image of a centaur correctly as having a torso and tail even though the attribute classifiers have never seen an image of a centaur. We use two methods for zero-shot learning: attribute-based recognition and direct-similarity based recognition. These two approaches differ in the way they describe unseen categories. For instance, an unseen category such as *leopard* will be described as living in **Africa** and being a member of the **feline** family by attribute-based recognition whereas direct similarity-based recognition will describe a leopard

as being similar to a **lion** and a **bobcat** in appearance.

Attribute-based recognition

For attribute-based recognition, we create a classifier per attribute, without using any example images from the testing categories. In order to label images of a category we rely on the text corpus and use the found category-attribute associations based on the attribute as in Section 3.3. During testing, we apply the attribute classifiers to a test image and obtain the probability of each attribute existing in the given test image, *i.e.* $p(a_1|x), p(a_2|x), \ldots$. The final classification of a test instance is performed using direct attribute prediction (DAP) proposed by Lampert *et al.* [45].

Using the DAP method, the posterior of a test category given a test image is calculated using:

$$p(z|x) = \sum_{a \in \{0,1\}^M} p(z|a)p(a|x) = \frac{p(z)}{p(a^z)} \prod_{m=1}^{M} p\left(a_m^z|x\right) \tag{3.7}$$

where z is the test category, a^z is the category-attribute associations for the category and M is the number of attributes. During testing, identical category priors $p(z)$ are assumed for each category. $p(a)$ is computed using a factorial distribution, $p(a) = \prod_{m=1}^{M} p(a_m)$ where the attribute priors are approximated using empirical means over the training categories, $p(a_m) = \frac{1}{K} \sum_{k=1}^{K} a_m^{y_k}$. This leads to the following MAP prediction f, that assigns the best output category from all test categories $z_1, \ldots, z_L$ to a test image x:

$$f(x) = \arg\max_{l=1,\ldots,L} \prod_{m=1}^{M} \frac{p(a_m^{z_l}|x)}{p(a_m^{z_l})} \tag{3.8}$$

Direct similarity-based recognition

For direct similarity-based recognition[65], we first train classifiers to separate each training category from others. Next, the semantic similarity between each testing and training category is computed. Using direct similarity, the posterior of a test

category given a test image is calculated using:

$$p(z|x) = \prod_{k=1}^{K} \left(\frac{p(y_k|x)}{p(y_k)} \right)^{y_k^z} \tag{3.9}$$

where $p(y_k|x)$ is the likelihood of a test image belonging to the training category y_k, and y_k^z is the computed semantic similarity between y_k and the testing category z. This is essentially applying the DAP method with $M = K$ (40 in case of the AwA dataset) attributes where each attribute classifier is trained using the instances of a single training category.

In order to compute the semantic similarity between the training and testing categories, we use the computed weight vectors of categories in Section 3.4.1 using:

$$y_k^z = \frac{\langle \vec{w_k}, \vec{w_z} \rangle}{\|\vec{w_k}\| \, \|\vec{w_z}\|} \tag{3.10}$$

3.6 Experiments

We performed experiments to analyze the success of the proposed methods at selecting attributes for plant and animal identification tasks in Section 3.6.1. Next, we use the attributes selected by the best candidate selection method in object recognition tasks in Section 3.6.2: We then group the selected attributes with respect to their semantics in Section 3.6.3. We use the selected attributes for supervised attribute-based classification of plants and we perform zero-shot learning of animals. Below, we explain these experiments in detail.

3.6.1 Comparison of methods for attribute selection

We compare four different strategies for proposing candidate words as visual attributes:

- *Method-1*: Iteratively selecting most frequently occurring words in the dictionary, as in [7], which is our baseline.

- *Method-2*: Estimating word weights using constraints only on the species (category) level and selecting words with the highest weights iteratively. This corresponds to applying the approach from Section 3.4.1, using only the first constraint in Equation 3.1.

- *Method-3*: Estimating word weights using taxonomy constraints and selecting words with the highest weights iteratively. This corresponds to applying the approach from Section 3.4.1, but not applying belief propagation across the word dendrogram.

- *Method-4*: Treating visual quality as a latent variable, and applying belief propagation across the automatically-constructed word dendrogram, as described in Section 3.4.2.

These methods are evaluated in terms of precision at selecting visual attributes and the resulting recognition accuracy, for the plant and animal identification tasks.

While *method-1* and *method-2* do not require any information other than textual documents describing categories, *method-3* is applicable when a taxonomy defined over the categories is available and *method-4* requires a word dendrogram to be constructed over the words in the text corpus. In our experiments, we use the weights learned by *method-3* to initialize the belief propagation procedure of *method-4*. However, *method-4* can also be used without a taxonomy, by using *method-2* to initialize the belief propagation. In summary, among the compared word selection strategies, methods 1, 2 and 4 are viable options in the absence of a taxonomy on categories.

Comparison of precisions

We compare the word selection strategies based on the number of candidates they need to assess in order to acquire a fixed number of visual attributes. In Figure 3-6 the performance of each word selection strategy at finding visual attributes is illustrated. The x axis in the figure is the number of candidate attributes that are assessed, and the y axis is the number of the visual attributes among the candidates. Various points on this figure are also presented in Table 3.1 for comparison. For instance

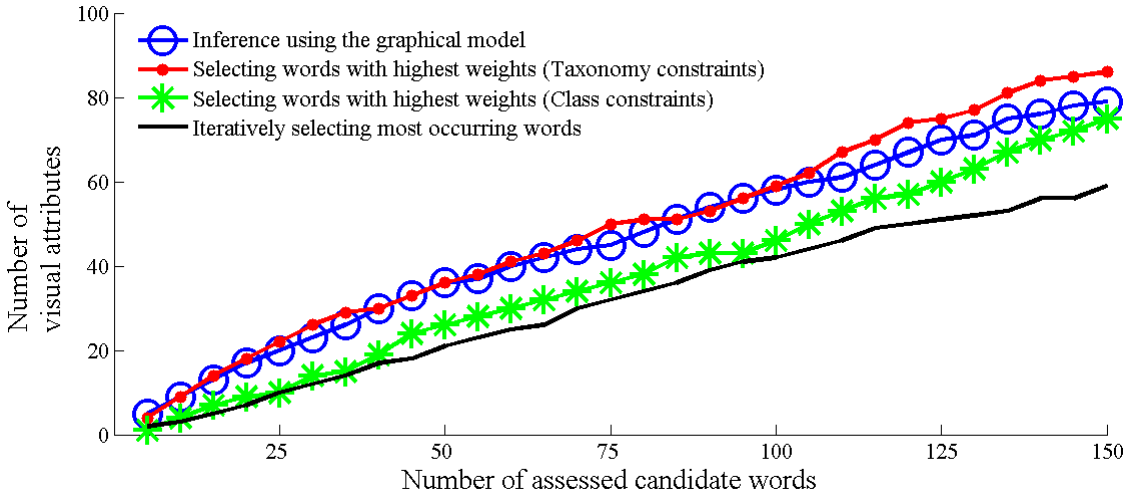


Figure 3-6: The comparison of number of visual attributes as a function of the number of candidates using various candidate word selection strategies for the plant identification task. All candidate selection strategies perform better than our baseline (black line) of iteratively selecting the most occurring words.

Table 3.1: Comparison of the number of candidates required to select M attributes (Smaller is better). The smallest number of candidates for each M is highlighted in bold.

	Candidate word selection methods			
	Method-1	*Method-2*	*Method-3*	*Method-4*
$M = 20$	45	41	**32**	35
$M = 40$	80	80	69	**68**
$M = 60$	130	115	**102**	104

using *method-4*, in order to obtain 60 visual attributes, 104 candidates need to be assessed.

We have used the output of the visual quality assessment as visualness ground-truth similar to [64, 20, 7]. The reason for this is threefold: (i) Humans/experts do not have a clear agreement as to which attributes are visual, (ii) As we are looking for visual attributes that are useful for recognition, the decision becomes even more complicated, and (iii) We found that the output of the proposed visual quality assessment correlates with human labels.

We compare the four candidate word selection methods in terms of the number of required candidates in the animal identification task for each feature descriptor.

Table 3.2: Comparison of the number of candidates required to select 25 visual attributes on the AwA dataset for each word selection strategy and for each provided feature descriptor (Smaller is better). The smallest number of candidates for each feature descriptor is highlighted in bold.

Feature descriptor	Candidate word selection methods			
	Method-1	*Method-2*	*Method-3*	*Method-4*
Color histogram	45	33	**31**	35
Local self similarity histogram	43	35	**33**	**33**
Histogram of oriented gradients	46	35	**33**	41
rgSIFT	46	35	**33**	36
SIFT	51	45	**33**	35
SURF	49	40	40	**36**

Specifically, we require each method to select 25 visual attributes and compare total number of assessed candidates for each method. The attribute selection results are presented in Table 3.2. For instance, using the SIFT descriptors and *method-4*, 35 candidates are assessed for selecting 25 visual attributes.

Compared to the baseline method of selecting the most occurring words in the text corpus (*method*-1), using *method*-2 (category-level constraints for learning the weights of words) results in faster, *i.e.* more precise, mining of visual attributes. In our experiments, for both animal and plant identification tasks, *method*-2 is favorable to *method*-1. Thus, we conclude that using category level constraints is useful for automatic attribute selection.

By integrating taxonomy constraints over *method*-2, we observe a significant performance gain using *method*-3. In fact, *method*-3 performs the best in terms of speed at selecting visual attributes in most of our experiments. These results show that having constraints using the taxonomy is crucial to identify candidate words that are likely to be visual attributes.

Adaptive word selection using belief propagation, *method*-4, builds upon *method*-

Table 3.3: The recognition accuracies of the evaluated methods for plant and animal identification.

	Candidate word selection methods			
	Method-1	*Method-2*	*Method-3*	*Method-4*
Plant Identification (%)	53.0	52.5	54.2	**54.6**
Animal Identification (%)	26.8	27.1	28.9	**30.4**

3 and incorporates information about word semantics and visual quality assessment into the word selection procedure, through the dendrogram of word similarity. We observe that, while performing competitively, *method*-4 fails to improve over *method*-3 in terms of speed in most of our experiments. This might be due to the fact that the graphical model needs to explore more words before exploiting the information about word semantics and visual quality assessment results.

Assessing a single candidate attribute (cross validation to find SVM parameters, training the classifier and testing on the evaluation set) on the ImageClef dataset takes around a minute while it takes around nine minutes on the AwA dataset, using a computer having Intel i7 2.00 GHz processor. *Method-4* requires the assessment of 216/104 candidates before finding the 25(*6)/60 attributes for the animal and plant identification tasks, respectively. On the other hand, *method-1* requires assessment of 280/130 candidates for the same task. So, if *method-4* is used to mine visual attributes instead of *method-1*, the total time gained for the plant identification task is 26 min whereas it is 576 min for the animal identification task. These time gains are significant even for relatively small datasets such as AwA and ImageClef.

Comparison of recognition accuracies

After the attributes are selected, we use the visual attributes in classification experiments. In this section we compare the attribute-based recognition accuracies of word selection strategies and in Section 3.6.2 we compare a word selection strategy with state of the art methods.

The resulting recognition accuracies for the plant and animal identification tasks are presented in Table 3.3. The plant and animal identification tasks are super-

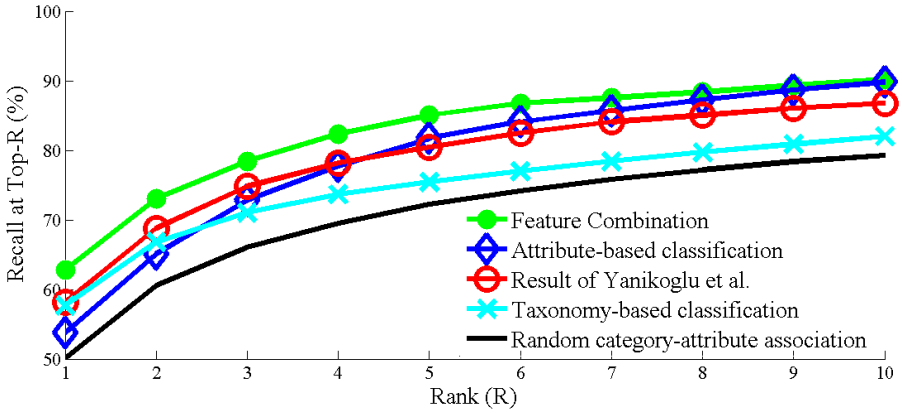


Figure 3-7: Comparison of the recognition accuracies of the tested methods on the ImageClef'2012 dataset, for the plant identification task at various ranks.

vised attribute-based classification (see Section 3.5.1) and zero-shot learning (see Section 3.5.2) tasks respectively. We see that the recognition accuracy for both plant and animal identification is the best when using the attributes selected by *method*-4.

In summary, we conclude that *method*-4 selects the best attributes in terms of recognition accuracy while also performing competitively in terms of precision. As we note note above, we think that the lower precision of *method*-4 with respect to *method*-3 might be due to the initial smoothing of the learned weights and the number of assessed candidates required to be able to propagate the evidence acquired from visual quality assessment. However, since *method*-4 also takes into consideration the distributional similarity of words, the visual attributes mined by *method*-4 are of higher quality for attribute-based recognition purposes.

3.6.2 Classification experiments

Below, we use the attributes discovered by *method*-4 for attribute-based recognition experiments and the weight vectors leaned by *method*-3 in direct similarity-based recognition experiments. We compare the proposed methods with the state-of-the-art methods in the two recognition tasks.

Supervised attribute-based classification for plant identification

We use the selected attributes to perform supervised attribute-based classification of plants. In the plant identification task, retrieval performance of a system is also important, so we present recalls for varying values of rank in Figure 3-7. For a specific rank, R, a classification decision is accepted as valid if the correct label is in the first Top-R guesses.

We compared five methods in the plant recognition experiment:

1. Supervised attribute-based classification described in Section 3.5.1.
2. The system of Yanikoglu *et al.* [83, 84] which had the best results at the ImageClef'2012 plant identification task.
3. Combination of the first two methods at feature level.
4. Taxonomy-based classification where given a plant taxonomic hierarchy with M nodes train M one-versus-all classifiers.
5. Randomly creating a category-attribute association matrix and using it for attribute-based recognition.

The results of the combined system are produced by an SVM classifier that is trained and tested on the concatenation of the attribute-based features (of length 60) with the features used by Yanikoglu *et al.* (of length 142) for each instance. The taxonomy-based classification approach creates classifiers to separate each species, genus and family from others in the taxonomic classification of plants to perform recognition. In total we created 235 classifiers for taxonomy-based classification. In our final experiment we randomly create a category-attribute association matrix with 60 attributes and train classifiers for each attribute. We repeat the same experiment 5 times and present the mean accuracy which is our baseline.

According to the results, randomly creating a category-attribute association matrix to perform attribute-based classification yields the lowest accuracies as expected. Using the taxonomic hierarchy of plants directly without using a text corpus improves

over the baseline. However, this method requires the training of more classifiers (235 compared to 60), does not create semantically meaningful attributes and fails to generalize as well the proposed method. Attribute-based classification using the attributes and the corresponding category-attribute associations we mine not only improves over the baseline it also generalizes well to the recognition task. For $R > 4$, the recognition results we obtain using attributes are better than the results Yanikoglu *et al.* obtain. Another observation is that the combined system performs the best for all R suggesting that attribute-based features complement low-level features for the plant identification task.

Zero-shot learning on Animals with Attributes (AwA) dataset

We illustrate the recognition performance of our system for animal identification using the AwA dataset. Lampert *et al.* [45] provide 6 feature descriptors for this database and we computed category-attribute associations for each category and for each descriptor. Next, for each category, we concatenated the category-attribute associations of each descriptor to create an extended representation. Thus, after combining all feature descriptors, each category has a representation of length $150 (= 25 * 6)$ during attribute-based recognition experiments. For direct similarity-based recognition experiments, we use the weight vectors of categories to compute semantic similarity between training and testing categories.

As an extension, in addition to the provided 6 feature descriptors of images, we perform recognition experiments using a new feature descriptor. Specifically, we extract a 4096-dimensional feature vector from each image using the Caffe [39] implementation of the convolutional neural network (CNN) described by Krizhevsky *et al.* [41]. These features are computed by forward propagating a mean-subtracted 227×227 RGB image through five convolutional layers and two fully connected layers. Using these state-of-the art features for attribute-based recognition we discover 50 attributes and perform the recognition experiment. During direct similarity-based recognition experiment, we train the classifiers of each category using the new feature descriptors.

10

Table 3.4: Comparison of zero-shot learning accuracies using attributes and direct similarity.

Author & knowledge base	Attribute selection	Category-attribute association	Accuracy (in %)
1. Attribute-based recognition			
Lampert *et al.* [44]	Manual	Manual	42.2
Rohrbach *et al.* [65] Wikipedia	Manual	Automatic	27.0
Rohrbach *et al.* [65] Wikipedia	Automatic	Automatic	19.7
Rohrbach *et al.* [65] Yahoo Img	Automatic	Automatic	23.6
Our method (Provided features)	Automatic	Automatic	30.4
Our method (CNN features)	Automatic	Automatic	45.7
2. Human supervision to compute similarity matrix			
Yu *et al.* [86] 85 attributes	Automatic	Automatic	42.3
Yu *et al.* [86] 200 attributes	Automatic	Automatic	48.3
3. Direct similarity-based recognition			
Rohrbach *et al.* [65] Wikipedia	Automatic	Automatic	33.2
Rohrbach *et al.* [65] Yahoo Img	Automatic	Automatic	35.7
Our method (Provided features)	Automatic	Automatic	41.8
Our method (CNN features)	Automatic	Automatic	59.4

We compare our results with the results of Lampert *et al.* [44], Yu *et al.* [86] and Rohrbach *et al.* [65] as presented in Table 3.4. While Lampert *et al.* use the manually defined 85 attributes and category associations in their experiments, Yu *et al.* utilize a similarity matrix created with human supervision to design attributes, and

[10]The extracted CNN features for the Awa dataset are available online at: https://drive.google.com/file/d/0Bx-64dmWqUHIVTdwS1QyTXlMT3M/view?usp=sharing

Rohrbach *et al.* present experiments with the manually defined attributes and mined category associations, with 74 mined attributes and corresponding category associations , and using direct similarity on several knowledge bases. We use 25 attributes (per feature descriptor) using the provided image descriptors, and 50 attributes using the features extracted by the CNN for attribute-based recognition experiments. In order to perform the direct similarity based recognition experiment with the provided feature descriptors, we concatenate the individual descriptors while training the classifiers of training categories. The comparison includes the knowledge base for the best performing strategy as well the results of Rohrbach *et al.* using Wikipedia as the knowledge base. Comparison with Wikipedia as the knowledge base is important since we also rely on encyclopedic information for attribute selection.

The highest recognition accuracies using the provided feature-set for attribute-based recognition are obtained by using manually defined attributes and category associations followed by our approach. Our method automatically discovers the attributes and the category-attribute associations while the accuracy we obtain surpasses the case where the attributes are defined manually and only the category-attribute associations are mined. This shows the superiority of our approach and the importance of using the taxonomy/distributional similarity information for automatic attribute selection. Furthermore, when using the CNN features in the attribute-based recognition experiment, a relative increase of around 50% in the recognition accuracy is achieved with respect to using the provided descriptors resulting in a zero-shot recognition accuracy of 45.7%.

In the direct similarity-based recognition experiment, using the provided features, our method obtains an accuracy of 41.8% which is higher than the accuracy obtained by Rohrbach *et al.* in their experiments. In this setting, our method gets a comparable performance to the accuracies obtained by Lampert *et al.* and Yu *et al.* with 85 attributes. Yu *et al.* report that the accuracy they obtain can be increased up to 48.3% by using more attributes. Using direct similarity and the CNN features, similar to the case of attribute based-recognition, the recognition accuracy we obtain increases up to 59.4%. To our knowledge, this is the best reported zero-shot recognition accuracy

Table 3.5: Selected attributes for plant identification.

Semantic category	Visual attributes
Groupings of plants	Sorbus, Acer, nutlet, maple, cernuous, Prunus, hawthorn, samara, mulberry, acorn, sumac, alder, birch, poplar, beech, pod
Context for plants	catkin, drupe, hypanthium, gland, bract, corymb, branchlet, corolla, floret
Visual qualities of plants	lobe, rounded, opposite, yellowgreen, yellow, white, glabrous, purple, serrate, vein, egg-shaped, conic, lanceolate, cordate, ovate, reddishbrown, chapped, pubescent, black, cusp, obovate, entire
False positives	female, root, centimeter, half(a), equal, length, narrowly, minutely, bear, rate, capsule, touch, fragrant

on this dataset.

3.6.3 The selected attributes

We present the 60 attributes discovered by *method*-4 for plant identification task in Table 3.5. We divide the words into 4 groups based on their semantics. The table contains words that may be grouped as:

1. Groupings of plants that are similar to each other such as *"Acer"*, *"Prunus"*, and *"Sorbus"*.

2. Context for plants such as *"catkin"*, *"gland"*, and "*branchlet"*.

3. Visual qualities of plants such as *"rounded"*, *"yellow"*, and *"lobe"*.

4. False positives such as *"female"*, *"root"*, and *"centimeter"*.

Note that words that are categorized as groupings of plants, parts of plants and visual qualities indeed refer to attributes that are used in object description by humans/experts. As for the words that are labeled as false positives, we included all

Table 3.6: Selected attributes for animal identification.

Semantic category	Visual attributes
Groupings of animals	ungulate, cetacean, canine, feline, kitten, cub, shark, jackal
Context for animals	Africa, graze, India, hunt, Waters, Indian, Pacific, sea, Atlantic, ocean, marine, Soviet, livestock, agricultural
Visual qualities of animals	hoof, ivory, giant, fancy
False positives	weightkg, jump, subspecies, trophy, disposition, favored, recover, university, imperativeness, estes, company, prey, tip, production, national, genome, standard, breed, intelligence, originally, owner, sizecm, engender, monk

words that we could not directly relate to visual attributes useful for object recognition, including generic words (*e.g.*, *"length", "minutely", "centimeter"*) and some others that may only be indirectly correlated with visual features (*e.g.*, *"fragrant"*).

The set of 50 attributes that are selected during our experiments using CNN features on the AwA dataset are presented in Table 3.6. The table contains the union of attributes that are selected by anyone of the descriptors given in Table 3.2. As before, we divided the visual words into 4 semantic categories that we relate to:

1. Groupings of animals such as *"cetacean", "feline"*, and *"shark"*.

2. Context for animals such as *"Africa", "agricultural"*, and ‘‘*sea"*.

3. Visual qualities of animals such as *"hoof", "ivory"*, and *"fancy"*.

4. False positives such as *"intelligence", "favored"*, *"weightkg"*.

We remind that we use the word "visual attribute" to refer to any word that may help in object recognition from images as described in Section 3.3. Indeed, we demonstrate the effectiveness of the selected visual attributes in object recognition, even though only a portion of them relate to truly visual qualities.

3.7 Conclusion and discussion

In this chapter, we tackle the important problem of automatically mining words referring to visual attributes. Our method assists the laborious attribute selection process and allows us to rapidly apply attribute-centric recognition to various recognition tasks. In order to mine attributes, we use the taxonomy of the domain, sample images and textual descriptions of object categories. We show the utility of our approach in two tasks: plant identification and zero-shot learning of animals.

Our contributions include two novel methods for identifying plausible candidates that can be used as attributes. While the first method uses the taxonomy defined on the categories and promotes candidates that conform to the taxonomy, the second method combines taxonomy with a distributional similarity measure. By providing a way to assess the visual quality of candidate words, we create an automatic system that generates a set of attributes for plant and animal identification tasks.

During experiments, we illustrate the utility of integrating taxonomy constraints for candidate word selection via two cases. In the first case we use only species (category) level constraints while in the second one we include taxonomy (species, genus and family) constraints. We show that taking advantage of the taxonomy yields substantially better results. The experiments also show that taxonomy-based distributional similarity of words can be used as a cue for selecting candidate words. By performing inference using the graphical model built on word dendrogram, we can further improve the recognition accuracies.

Plant/animal identification are both challenging problems and we demonstrate the usefulness of the mined set of attributes by using the trained attribute classifiers in both of these tasks. In the plant identification task the attribute classifiers are used for feature extraction in a supervised learning setting; while in animal identification, they are used for zero-shot learning of unseen animal categories.

During the plant identification experiments, we show that using attribute-based features bring performance improvements compared to using only lower-level features. The classification results also highlight better performance of attribute-based

features with increasing ranks, while providing a compact representation. The zero-shot learning of animals demonstrate the quality of our attribute selection procedure: our system that automatically selects both attributes and category-attribute associations, achieves better recognition results than the state-of-the-art results obtained with manually defined attributes and mined category-attribute associations, in the same dataset. Using direct similarity-based recognition, we further improve our results and obtain recognition accuracies comparable to the case where both attributes and category-attribute associations are selected manually. In the recognition experiments, we also present the performance of our system using state-of-the-art CNN features and, to our knowledge, get the best zero-shot recognition accuracies obtained on the AwA dataset.

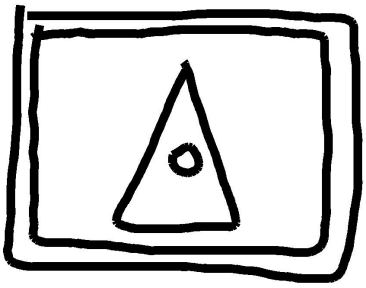
Chapter 4

User interfaces

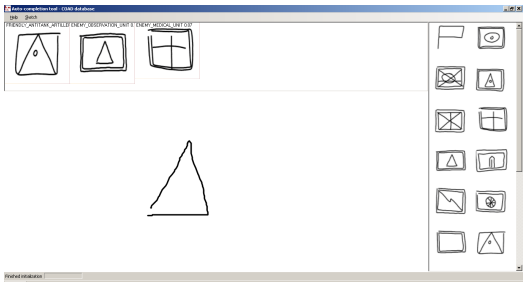
In the previous chapters we introduced algorithms that provide auto-completion of sketched symbols and automatically mine attributes and category-attribute associations for object recognition from images. During experiments, we showed the utility of the proposed approaches in terms of recognition accuracy. In this section, we examine auto-completion and attribute-based learning from a human-computer interaction perspective and demonstrate two applications that utilize the proposed algorithms to provide natural and efficient query interfaces.

We believe that both auto-completion and attribute-centric recognition have the potential to facilitate the way humans interact with computers. Sketching is the most natural form of interface for certain applications and auto-completion of sketches can make it more efficient for users to interact with the computer. On the other hand, attributes allow a way for humans to communicate with computers using semantically meaningful words, especially in applications where the user is searching, querying or browsing images. Attribute-based search and browsing interfaces thus provide a natural alternative to text or image-based queries that are not very effective. For instance, searching for an unknown plant using image search (called reverse image search in Google) is currently not very successful; similarly text-based search in this application is not at all suitable.

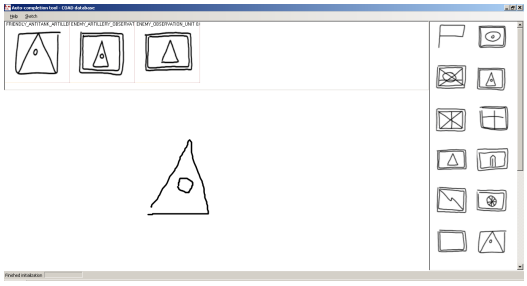
In the following sections, we present prototype interfaces that realize sketched-symbol recognition with auto completion and attribute-centric search among plant



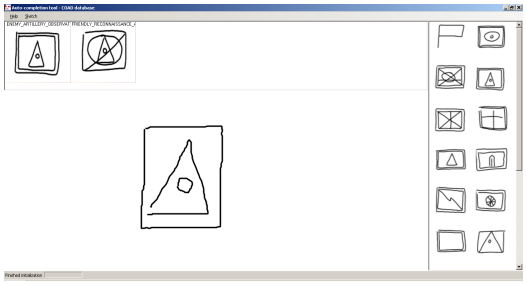
(a) The intended symbol



(b) Output after the first stroke



(c) Output after the second stroke



(d) Output after the third stroke

Figure 4-1: Screenshots of the auto-completion application interface. The application predicts what the user intends to draw for the COAD database. Thanks to auto-completion, users can quickly finish drawing before drawing symbols in their entirety.

categories. Although we do not have user experiments regarding how the ideas developed in the thesis affect the quality the way users sketch or search images, we believe these interfaces are important to establish a proof of concept.

4.1 Auto-completion application

The auto-completion application [1] automatically guesses as to what the user intends to draw with each stroke for the COAD database. The number of guesses the application makes is determined by the confidence of the auto-completion algorithm. The application provides the user with at most three guesses or until the confidence is higher than 0.9. For instance, if the posterior probabilities of the most likely three

[1] publicly available online at https://www.dropbox.com/sh/qpvf6o8o2cwa6oy/AADufXiU6n6nSJDmvvGrm_Tga?dl=0

symbol categories are 0.4, 0.3 and 0.25 then all three symbol categories are presented to the user. On the other hand, if the probabilities are 0.6, 0.32 and 0.03 only the first two categories are presented to the user. In the application the auto-completion algorithm uses the CKMeans clustering algorithm where the number of clusters is set to 40 since we obtained the best test results using this setting. Using the application, users not only familiarize themselves with the symbols quickly but also complete what they intend to draw faster. We also provide the confidence the algorithm has in each predicted symbol category in the form of tooltips. By moving the cursor on the presented classification results the user can observe how the confidence in prediction changes with each stroke.

The auto-completion application is illustrated in Figure 4-1 with an example. Suppose that the user intends to draw the symbol given in Figure 4-1a. The user starts by drawing the inner triangle presented in Figure 4-1b. The auto-completion application makes three guesses but the intended symbol is not among them. The main reason for that is the high level of ambiguity, *i.e.* there is little information in the current state to determine the intended symbol. Still, the first two candidates that the algorithm presents contain triangles so they are actually valid candidates. Next, the user follows by drawing the next stroke and draws a circle inside the triangle as presented in Figure 4-1c. This time, the intended symbol is in the second guess of the algorithm and the user can quickly finish drawing by selecting the relevant symbol. Thus, instead of using 4 strokes to draw the symbol the user can actually finish drawing with only 2 strokes. If the user draws the third stroke to produce the state given in Figure 4-1d, the algorithm is more confident as to which symbols are possible. The algorithm presents the user with two candidates where the intended symbol is the most likely candidate.

More screenshots of the auto-completion application are presented in Figure 4-2. As can be observed, auto-completion significantly reduces the effort of the user for the COAD symbols, *i.e.* the user can finish drawing with fewer strokes. At the same time, since the user is able to see possible candidates, it is easier for the user to familiarize with the symbols. Thus, auto-completion application is also useful to reduce the time

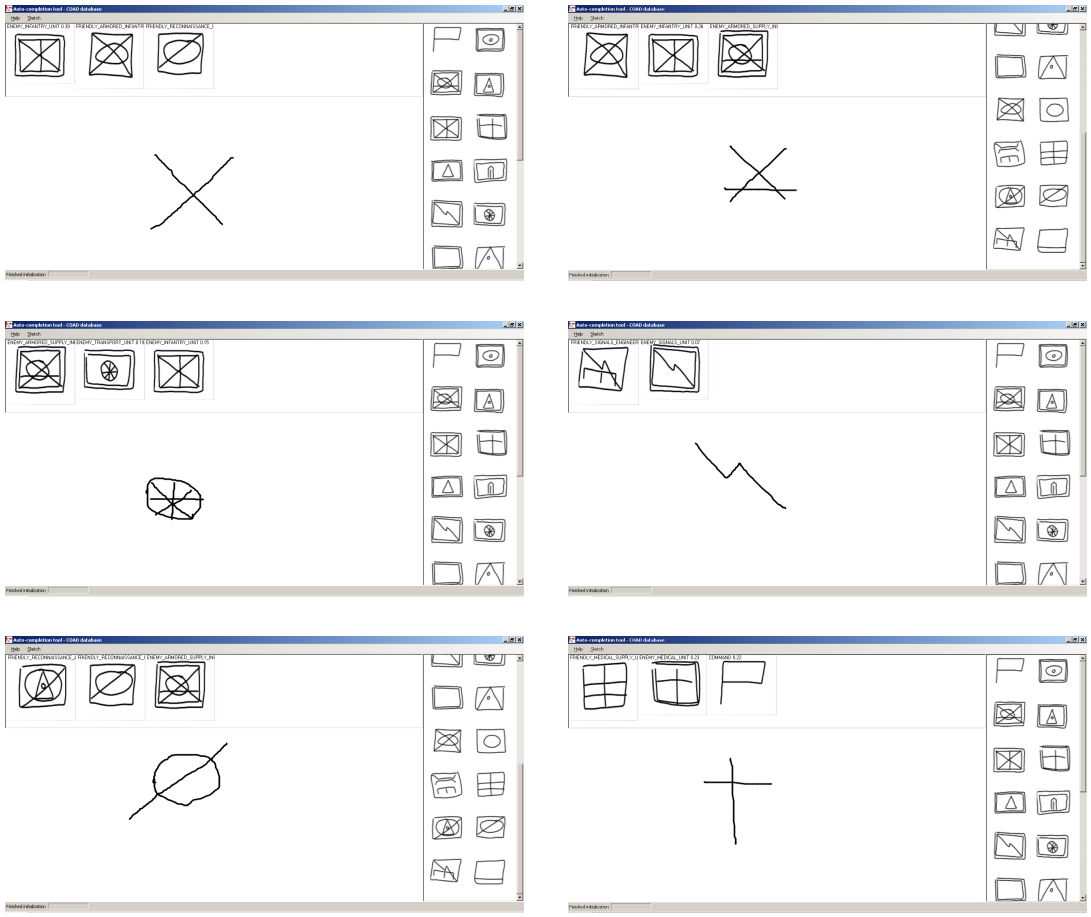


Figure 4-2: Various examples of auto-completion are presented in the images. As can be observed, auto-completion helps to reduce the number of strokes required to draw sketched symbols significantly for the symbols in the COAD dataset.

it takes for the users to become proficient with sketching interfaces.

4.2 Attribute-based image search application

The attribute-based search application [2] provides a way to search for plant categories through attributes. The application asks the user a series of questions in the form of existence of attributes and retrieves the most likely plant categories according to the answers. This interaction offers a convenient and fast way for the users to search through plant categories.

In the application, we use all the attributes and corresponding category-attribute associations mined for the plant identification task except the attributes that are marked as false positives in Table 3.5. That is, we use the mined attributes that represent groupings of plant categories, contextual information and visual qualities along with their category-attribute associations in the application.

The main algorithm of the attribute-based image search is presented in Algorithm 2. The application starts by asking the user a question (line 6) for which the user gives one of the three following answers (line 7): *yes, no, pass*. The answers *yes, no* and *pass* stand for thoughts of the user about the attribute existing in the searched image. The answers *yes* and *no* stand for the current attribute existing and not existing in the image respectively while *pass* is selected if the user cannot make a guess or if the attribute is ambiguous. With each user answer, the predictions are updated (line 8) and the user is presented with the most likely categories. The predictions are made using the attribute-based zero-shot learning algorithm discussed in Section 3.5.2 and the probability of each plant category are updated.

In order to select the question to ask the user, the application uses Algorithm 3. Firstly, the remaining attributes (line 2) and the categories satisfying previous answers (line 3) are selected. The categories satisfying previous answers are found using the category-attribute association matrix where the categories that are in line with the answers of the user are selected. For each remaining attribute, information-gains

[2] publicly available online at https://github.com/mustafae/20qs

Algorithm 2 Main procedure for attribute-based image search.

```
1: function IMAGE SEARCH
2:     Previous_Answers ← ∅
3:     Previous_Attributes ← ∅
4:     Initialize predictions with each category having probability
5:     while the user cannot find the searched image category do
6:         Select Next_Attribute Algorithm 3
7:         Next_Answer ← The answer of the user
8:         Use Attribute-based zero-shot learning to update predictions
9:         Previous_Attributes ← Previous_Attributes ∪ Next_Attribute
10:        Previous_Answers ← Previous_Answers ∪ Next_Answer
11:    end while
12: end function
```

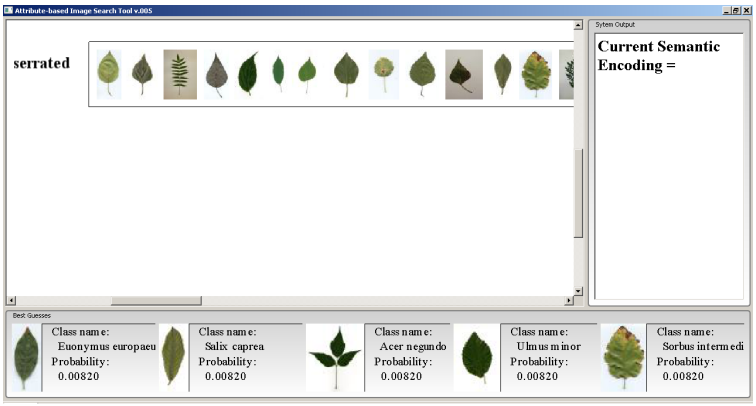
Algorithm 3 Procedure for attribute selection.

```
1: function ATTRIBUTE SELECTION(Previous_Answers, Previous_Attributes)
2:     Remaining_Attributes ← Attributes \ Previous_Attributes
3:     Remaining_Categories ← Categories that satisfy Previous_Answers for Previous_Attributes
4:     Information_Gains ← The information gain of Remaining_Attributes on the Remaining_Categories
5:     return arg max_{Remaining_Attributes} Information_Gains
6: end function
```

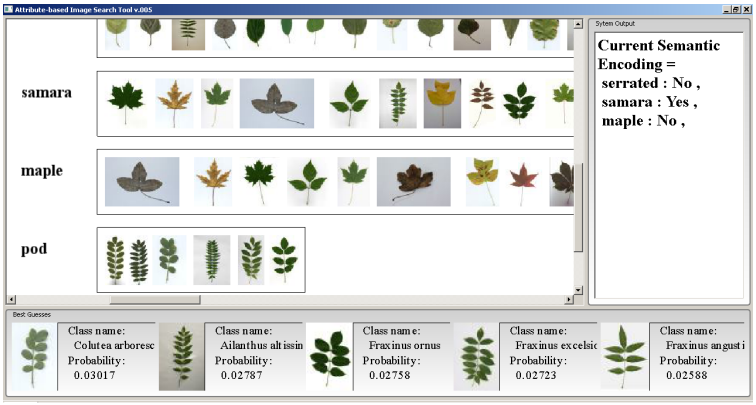
are computed (line 4) using the category-attribute associations. The attribute that maximizes the information gain is selected as the next question (line 5).

We illustrate the attribute-based image search in Figure 4-3. Suppose that the user is is searching an unknown plant from the Fraxinus excelsior species, presented in Figure 4-3a. The initial state of the application is presented in Figure 4-3b. The top pane is used to ask questions and to present images that satisfy attributes, *e.g.*, in the initial state existence of the attribute *serrated* is asked to the user while images satisfying the attribute are presented to its right. The right pane is used to keep track of the user answers which is initially empty. As the user search progresses the right pane is filled with the answers of the user. The bottom pane is used to present the top 5 most likely candidates for the category the user is searching. After each time the user answers a question with a *yes* or *no* the bottom pane is updated with the new predictions. In this example, the user is able to find the searched category by answering only 3 questions about existence of attributes as presented in Figure 4-3c.

(a) The searched image category - Fraxinus excelsior



(b) The initial state of the application. All plant categories have equal probability initially. The user is first asked whether the queried plant is *serrated*.



(c) The user answered the first three questions *serrated*, *samara* and *maple* with *no*, *yes* and *no* respectively. The most likely 5 candidate categories are presented at the bottom of the application with the searched category displayed in the 4^{th} image from left. Thus, the user found the searched category after answering three questions about the existence of attributes.

Figure 4-3: The application allows users to search for plant categories using attributes. Thanks to attribute-based search, users can quickly find the category they are looking for by answering a few questions.

The attributes that are asked to the user in this example are *serrated*, *samara* and *maple* while the answers of the user are *no*, *yes* and *no* respectively.

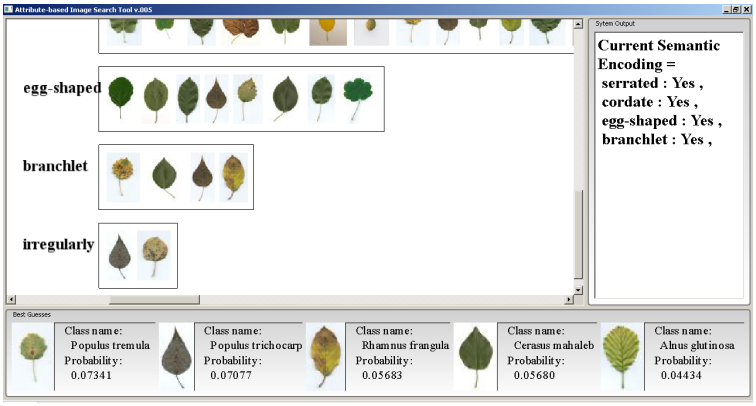
One concern with the attribute-based search application is that some of the attributes might be hard to understand/visualize. For instance, the plants having the attribute *serrated* have toothed leaf margins. However, it might be hard for the user to visualize the attribute by just looking at the representative images of the attribute

serrated. It would be nice if the application could highlight the parts of the images having the attribute so that the user could understand the meaning of *serrated* better. Furthermore, some of the attributes such as *maple* refer to global shape of leaves. If the application can convey that information to the user, it would be easier for the user to decide on the answer. With those features, even without domain expertise, a user might correctly answer whether or not an attribute exists just through visual inspection.

Another point of note is that, the results retrieved in Figure 4-3c look quite similar to each other and three of them are members of the Fraxinus genus. This is related to our assumption of closer species in the taxonomy being closer to each other visually. For this specific example, we can see that our assumption holds and the user can not only retrieve the target category but also its close relatives as well.

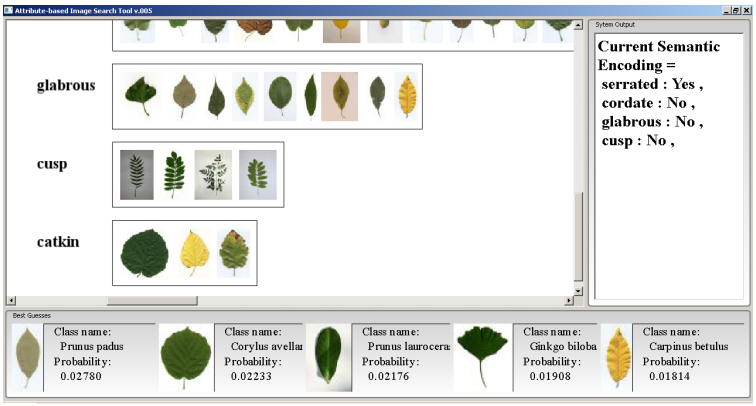
Further examples of attribute-based search are illustrated in Figure 4-4. As can be observed, attribute-based search allows searching efficiently and requires only a few steps before finding the searched plant category.

(a) The searched image category: Cerasus mahaleb



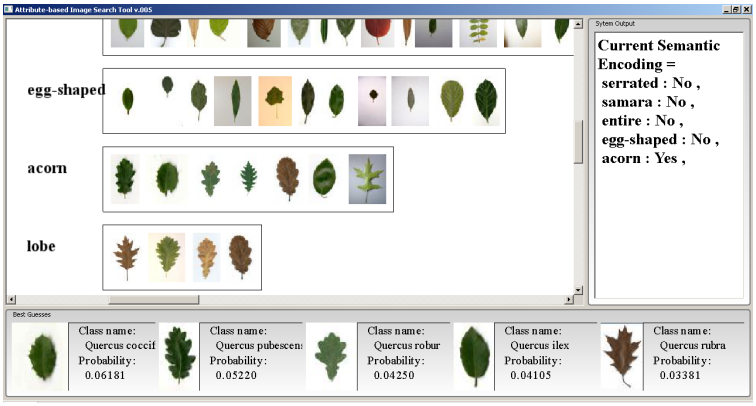
(b) The searched category is found after answering 4 questions *(serrated: yes, cordate: yes, egg-shaped: yes, branchlet: yes)*

(c) The searched image category: Gingko biloba



(d) The searched category is found after answering 4 questions *(serrated: yes, cordate: no, glabrous: no, cusp: no)*

(e) The searched image category: Quercus rubra



(f) The searched category is found after answering 5 questions *(serrated: no, samara: no, entire: no, egg-shaped: no, acorn: yes)*

Figure 4-4: Various examples of attribute-based search.

Chapter 5

Summary and future work

In the thesis, we have designed and implemented computer vision and machine learning algorithms that will help create more intuitive and user-friendly interfaces in two different domains. In Chapter 2, we discussed a method to realize auto completion of sketched symbols and show that auto-completion of sketched symbols can be performed in real-time with accuracies close to a human expert. In Chapter 3, we discussed how visual attributes can be mined from text automatically and a taxonomy over categories and word semantics can be utilized to improve attribute mining speed and quality. We showed the value of the mined attributes through experiments and improved the recognition accuracies in plant identification and zero-shot learning. In Chapter 4, we created prototype interfaces using the algorithms discussed in previous chapters in order to create more intuitive and user-friendly interfaces.

While we discussed auto-completion of sketched symbols, one direction for future work is to research how auto-completion affects the user when employed in free form drawings consisting of symbols. That is, when the user sketches several symbols and auto-completion is used to predict components, as they are being drawn. This way, while the user is restricted to work on one symbol at a time, the segmentation phase that is required in such recognition tasks can be eliminated.

Another research direction is user adaptation, which is studied in handwriting recognition. Auto-completion can adapt to the way a certain symbol is drawn by a user, but more importantly the preferred stroke order of the user can also be read-

ily incorporated into our algorithm. While our auto-completion system inherently leverages the fact that humans tend to prefer certain orderings of strokes over others, we did not perform user adaptation. Auto-completion performance can be increased greatly for a specific user if user adaptation is applied. This is due to the fact that individuals have a certain sketching/drawing style and they stick to their styles during sketching. If user adaptation is applied to auto-completion, the knowledge of individual sketching styles can be integrated into auto-completion leading to better auto-completion performance.

We used attribute-based learning for object recognition from images. Attribute-based learning can also be utilized for sketch recognition and auto-completion. An auto-completion system that uses attributes of sketches for prediction and auto-completion can be an interesting future work. For instance, each category can be associated with attributes and the existence of attributes could be used to predict probable categories.

Our automatic category-attribute association algorithm uses the existence of an attribute in the description of a category to signal the existence of the attribute for that category. Consider the word "lobes" as an attribute where the description of a category contains the following sentence: "...does not have lobes ...". In this example, although the word "lobes" is used in the description, it is used negatively. An NLP system that can extract such negations from textual data would improve both attribute selection speed and quality. Another way to improve attribute-based learning is to extract relations from textual data. Consider the word "yellow" in the descriptions of two plants: "...its leaves turn yellow in autumn ..." and "...has a yellow fruit ...". While the word "yellow" is used in the descriptions of both plants, in the former case "yellow" is used to refer to leaves and in the latter it is used to refer to fruits. If such relations can be extracted reliably it would be valuable for attribute selection. Both of theses improvements to attribute mining are possible with the usage of proper NLP algorithms.

In our discussion about the attribute-based image search application we restricted ourselves to the attributes we mined. However, it would be nice if the users could

extend the set of available attributes by defining their own. While this requires manual work, enabling such a feature might allow for a more intuitive and easy to understand image search interface.

Finally, both sketch recognition and attribute-based learning can benefit from recent improvements in machine learning, specifically the development of deep learning strategies. Using deep learning to predict sketched symbol categories and attributes can improve recognition accuracies in both domains.

Bibliography

[1] Christine Alvarado. Sketch recognition user interfaces: Guidelines for design and development. In *AAAI 2004 Symposium on Making Pen-Based Interaction Intelligent and Natural*, pages 8–14, Menlo Park, California, October 21-24 2004. AAAI Fall Symposium. 21

[2] Christine Alvarado and Randall Davis. Resolving ambiguities to create a natural sketch based interface. In *Proceedings of IJCAI*, pages 1365 – 1371, August 2001. 22

[3] Relja Arandjelović and Tevfik Metin Sezgin. Sketch recognition by fusion of temporal and image-based features. *Pattern Recognition*, 44:1225–1234, June 2011. 44

[4] James Arvo and Kevin Novins. Fluid sketches: Continuous recognition and morphing of simple hand-drawn shapes. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 73–80. ACM, 2000. 22

[5] Kobus Barnard and Keiji Yanai. Mutual information of words and pictures. In *Information Theory and Applications Inaugural Workshop*, February 2006. 51

[6] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:509–522, 2001. 55

[7] Tamara L. Berg, Alexander C. Berg, and Jonathan Shih. Automatic attribute discovery and characterization from noisy web data. In *Proceedings of the 11th European conference on Computer vision: Part I*, ECCV'10, pages 663–676, Berlin, Heidelberg, 2010. Springer-Verlag. 10, 48, 51, 54, 56, 68, 70

[8] Alexander Binder, Klaus-Robert Müller, and Motoaki Kawanabe. On taxonomies for multi-class image categorization. *International Journal of Computer Vision*, 99(3):281–301, 2012. 53

[9] William Blacoe and Mirella Lapata. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 546–556, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. 63

[10] A. Blessing, T. M. Sezgin, R. Arandjelovic, and P. Robinson. A multimodal interface for road design. In *Workshop on Sketch Recognition, International Conference on Intelligent User Interfaces*, 2009. 20

[11] Erik Boiy, Koen Deschacht, and Marie-Francine Moens. Learning visual entities and their visual attributes from text corpora. In *DEXA Workshops*, pages 48–53. IEEE Computer Society, 2008. 51

[12] Steve Branson, Catherine Wah, Florian Schroff, Boris Babenko, Peter Welinder, Pietro Perona, and Serge Belongie. Visual recognition with humans in the loop. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *ECCV (4)*, volume 6314 of *Lecture Notes in Computer Science*, pages 438–451. Springer, 2010. 48

[13] Florian Brieler and Mark Minas. A model-based recognition engine for sketched diagrams. *Journal of Visual Languages and Computing*, 21:81–97, April 2010. 20

[14] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. 43, 66

[15] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995. 29

[16] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, 2005. 55

[17] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977. 27

[18] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 52

[19] Jia Deng, Alexander C. Berg, Kai Li, and Li Fei-Fei. What does classifying more than 10,000 image categories tell us? In *Proceedings of the 11th European conference on Computer vision: Part V*, ECCV'10, pages 71–84, Berlin, Heidelberg, 2010. Springer-Verlag. 52

[20] Santosh K. Divvala, Ali Farhadi, and Carlos Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 10, 54, 56, 70

[21] Jesse Dodge, Amit Goyal, Xufeng Han, Alyssa Mensch, Margaret Mitchell, Karl Stratos, Kota Yamaguchi, Yejin Choi, Hal Daumé III, Alexander C. Berg, and Tamara L. Berg. Detecting visual text. In *HLT-NAACL*, pages 762–772. The Association for Computational Linguistics, 2012. 51

[22] Manual F. 101-5-1, Operational Terms and Graphics. *Washington, DC, Department of the Army 30*, 1997. 31

[23] Ali Farhadi, Ian Endres, and Derek Hoiem. Attribute-centric recognition for cross-category generalization. In *CVPR*, pages 2352–2359, 2010. 48

[24] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *CVPR*, pages 1778–1785. IEEE, 2009. 48

[25] Guihuan Feng, Christian Viard-Gaudin, and Zhengxing Sun. On-line hand-drawn electric circuit diagram recognition using 2d dynamic programming. *Pattern Recognition*, 42:3215–3223, December 2009. 20

[26] Yansong Feng and Mirella Lapata. Visual information in semantic representation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 91–99, Los Angeles, California, June 2010. Association for Computational Linguistics. 53

[27] Vittorio Ferrari and Andrew Zisserman. Learning visual attributes. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 433–440, Cambridge, MA, 2008. MIT Press. 48

[28] Andrea Frome, Yoram Singer, Fei Sha, and Jitendra Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *ICCV*, pages 1–8, 2007. 61

[29] Hervé Goëau, Pierre Bonnet, Alexis Joly, Itheri Yahiaoui, Daniel Barthelemy, Nozha Boujemaa, and Jean-François Molino. The ImageCLEF 2012 plant identification task. *CLEF 2012 working notes*, September 2012. 49

[30] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007. 52

[31] G. Griffin and P. Perona. Learning and using taxonomies for fast visual categorization. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008. 52

[32] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. In *SIGKDD Explorations*, volume 11, 2009. 27

[33] Tracy Hammond and Randall Davis. Automatically transforming symbolic shape descriptions for use in sketch recognition. In *Proceedings of the 19th national conference on Artifical intelligence*, AAAI'04, pages 450–456. AAAI Press, 2004. 20

[34] Tracy Hammond and Randall Davis. Ladder, a sketching language for user interface developers. *Computers and Graphics*, 29:518–532, August 2005. 20

[35] Zellig Harris. Distributional structure. *Word*, 10(23):146–162, 1954. 63

[36] Heloise Hse and A. Richard Newton. Sketched symbol recognition using zernike moments. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 1 - Volume 01*, ICPR '04, pages 367–370, Washington, DC, USA, 2004. IEEE Computer Society. 20

[37] Heloise Hwawen Hse and A. Richard Newton. Recognition and beautification of multi-stroke symbols in digital ink. *Comput. Graph.*, 29:533–546, August 2005. 22

[38] Takeo Igarashi, Satoshi Matsuoka, Sachiko Kawachiya, and Hidehiko Tanaka. Interactive beautification: a technique for rapid geometric design. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, UIST '97, pages 105–114, New York, NY, USA, 1997. ACM. 22

[39] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 75

[40] Levent B. Kara and Thomas F. Stahovich. Hierarchical parsing and recognition of hand-sketched diagrams. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 13–22, New York, NY USA, 2004. ACM. 20

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 75

[42] Neeraj Kumar, Alexander C. Berg, Peter N. Belhumeur, and Shree K. Nayar. Attribute and simile classifiers for face verification. In *ICCV*, pages 365–372. IEEE, 2009. 48

[43] Neeraj Kumar, Alexander C. Berg, Peter N. Belhumeur, and Shree K. Nayar. Describable visual attributes for face verification and image search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(10):1962–1977, 2011. 48

[44] C.H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(3):453–465, March 2014. 76

[45] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, pages 951–958. IEEE, 2009. 48, 50, 56, 67, 75

[46] James A. Landay and Brad A. Myers. Interactive sketching for the early stages of user interface design. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '95, pages 43–50, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. 22

[47] Junfeng Li, Xiwen Zhang, Xiang Ao, and Guozhong Dai. Sketch recognition with continuous feedback based on incremental intention extraction. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 145–150, New York, NY, USA, 2005. ACM. 22

[48] Peng Liu, Lei Ma, and Frank K. Soong. Prefix tree based auto-completion for convenient bi-modal chinese character input. In *ICASSP*, pages 4465–4468, 2008. 23

[49] A. Chris Long, Jr., James A. Landay, Lawrence A. Rowe, and Joseph Michiels. Visual similarity of pen gestures. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '00, pages 360–367, New York, NY, USA, 2000. ACM. 20

[50] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008. 60

[51] J. Mas, G. Sanchez, J. Llados, and B. Lamiroy. An incremental on-line parsing algorithm for recognizing sketching diagrams. In *ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition*, pages 452–456, Washington, DC, USA, 2007. IEEE Computer Society. 22

[52] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. 52

[53] Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1439, 2010. 63

[54] Ralph Niels, Don Willems, and Louis Vuurpijl. The NicIcon database of handwritten icons. In *11th International Conference on the Frontiers of Handwriting Recognition*, Montreal, Canada, August 2008. 32

[55] Michael Oltmans. *Envisioning sketch recognition: a local feature based approach to recognizing informal sketches*. PhD thesis, Cambridge, MA, USA, 2007. 20

[56] Tom Y. Ouyang and Randall Davis. A visual approach to sketched symbol recognition. In *Proc. International Joint Conferences on Artificial Intelligence*, pages 1463–1468, 2009. 26

[57] Devi Parikh. Discovering localized attributes for fine-grained recognition. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 3474–3481, Washington, DC, USA, 2012. IEEE Computer Society. 48, 51

[58] Devi Parikh and Kristen Grauman. Interactively building a discriminative vocabulary of nameable attributes. In *CVPR*, pages 1681–1688. IEEE, 2011. 48, 51

[59] Brandon Paulson and Tracy Hammond. A system for recognizing and beautifying low-level sketch shapes using ndde and dcr. 20th Annual ACM Symposium on User Interface Software and Technology Posters, October 2007. 22

[60] Brandon Paulson and Tracy Hammond. Paleosketch: accurate primitive sketch recognition and beautification. In *Proceedings of the 13th international conference on Intelligent user interfaces*, IUI '08, pages 1–10, New York, NY, USA, 2008. ACM. 22

[61] Theo Pavlidis and Christopher J. Van Wyk. An automatic beautifier for drawings and illustrations. *SIGGRAPH Comput. Graph.*, 19(3):225–234, 1985. 22

[62] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. 64

[63] Beryl Plimmer and John Grundy. Beautifying sketching-based design tool content: Issues and experiences. In *in 'Proceedings of the Sixth Australasian User Interface Conference (AUIC2005*, volume 40, pages 31–38, 2005. 22

[64] Mohammad Rastegari, Ali Farhadi, and David Forsyth. Attribute discovery via predictable discriminative binary codes. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI*, ECCV'12, pages 876–889, Berlin, Heidelberg, 2012. Springer-Verlag. 10, 51, 54, 70

[65] Marcus Rohrbach, Michael Stark, György Szarvas, Iryna Gurevych, and Bernt Schiele. What helps where - and why? semantic relatedness for knowledge transfer. In *CVPR*, pages 910–917. IEEE, 2010. 48, 51, 52, 67, 76

[66] Dean Rubine. Specifying gestures by example. *Computer graphics and interactive techniques*, 25:329–337, July 1991. 20

[67] Olga Russakovsky and Fei-Fei Li. Attribute learning in large-scale datasets. In Kiriakos N. Kutulakos, editor, *ECCV Workshops (1)*, volume 6553 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2010. 48

[68] S. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, August 2007. 63

[69] Tevfik Metin Sezgin. Sketch based interfaces: Early processing for sketch understanding. In *Proceedings of PUI-2001. NY*. ACM Press, 2001. 22

[70] Tevfik Metin Sezgin and Randall Davis. Sketch recognition in interspersed drawings using time-based graphical models. *Computers & Graphics*, 32(5), 2008. 26

[71] Carina Silberer, Vittorio Ferrari, and Mirella Lapata. Models of semantic representation with visual attributes. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 572–582, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. 53

[72] Saul Simhon and Gregory Dudek. Sketch interpretation and refinement using statistical models. In Alexander Keller and Henrik Wann Jensen, editors, *Rendering Techniques*, pages 23–32. Eurographics Association, 2004. 26

[73] Caglar Tirkaz, Jacob Eisenstein, T. Metin Sezgin, and Berrin Yanikoglu. Identifying visual attributes for object recognition from text and taxonomy. *Computer Vision and Image Understanding*, (0):–, 2015. 16

[74] Caglar Tirkaz, Berrin Yanikoglu, and T. Metin Sezgin. Sketched symbol recognition with auto-completion. *Pattern Recognition*, 45(11):3926 – 3937, 2012. 16

[75] R. S Tumen, M. E Acer, and T. M Sezgin. Feature extraction and classifier combination for image-based sketch recognition. *ACM Symposium on Sketch Based Interfaces and Modeling*, June 7-10, (2010). 37

[76] P. van Sommers. *Drawing and Cognition: Descriptive and Experimental Studies of Graphic Production Processes*. Cambridge University Press, 1984. 26

[77] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schroedl. Constrained k-means clustering with background knowledge. In *ICML*, pages 577–584. Morgan Kaufmann, 2001. 27

[78] Paul Wais, Aaron Wolin, and Christine Alvarado. Designing a sketch recognition front-end: user perception of interface elements. In *SBIM '07: Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, pages 99–106, New York, NY, USA, 2007. ACM. 21

[79] Gang Wang and David A. Forsyth. Joint learning of visual attributes, object classes and visual saliency. In *ICCV*, pages 537–544. IEEE, 2009. 48

[80] Josiah Wang, Katja Markert, and Mark Everingham. Learning models for object recognition from natural language descriptions. In *Proceedings of the British Machine Vision Conference*, pages 2.1–2.11. BMVA Press, 2009. doi:10.5244/C.23.2. 48

[81] Yang Wang and Greg Mori. A discriminative latent model of object classes and attributes. In *Proceedings of the 11th European conference on Computer vision: Part V*, ECCV'10, pages 155–168, Berlin, Heidelberg, 2010. Springer-Verlag. 48

[82] Don Willems, Ralph Niels, Marcel van Gerven, and Louis Vuurpijl. Iconic and multi-stroke gesture recognition. *Pattern Recognition*, 42(12):3303 – 3312, 2009. 20, 40

[83] Berrin A. Yanikoglu, Erchan Aptoula, and Caglar Tirkaz. Sabanci-okan system at imageclef 2012: Combining features and classifiers for plant identification. In *CLEF (Online Working Notes/Labs/Workshop)*, 2012. 74

[84] Berrin A. Yanikoglu, Erchan Aptoula, and Caglar Tirkaz. Automatic plant identification from photographs. *Mach. Vis. Appl.*, 25(6):1369–1383, 2014. 74

[85] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, 8:236–239, 2003. 64

[86] F.X. Yu, Liangliang Cao, R.S. Feris, J.R. Smith, and Shih-Fu Chang. Designing category-level attributes for discriminative visual recognition. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 771–778, June 2013. 52, 76