I give permission for public access to my thesis and for any copying to be done at the discretion of the archives librarian and/or the College librarian.

.....

June 3, 2005 Desislava Petkova

Abstract

The availability of digital image databases creates a demand for tools which automatically retrieve relevant images in response to user queries. A common approach to image retrieval is to use annotations as image proxies and to compare the query words and the description words of candidate images. This technique requires that annotations are produced in advance. However, for collections of realistic size, it is inconceivable to rely exclusively on manual annotation. A practical alternative is automatic annotation, where a computer system analyzes the visual features of an image to determine the appropriate description.

A system for automatic image annotation learns from manually annotated images what correlations exist between words and visual components, and then uses the discovered relationships to automatically assign semantically descriptive words to new, unannotated images. Our work builds on a cross-media relevance modeling approach which learns such correspondences by comparing the visual components of the image to be annotated with those of already annotated images. We extend the relevance model with clustering, a technique for partitioning a space into groups of similar objects. The motivation is that clustering utilizes collection-wide features, which are ignored by an individual-document analysis, and that the similarity structure of the collection as a whole is a potentially useful source of information.

We propose and evaluate two cluster-based relevance models. We compare their performance with that of the baseline unclustered model and show that by using cluster statistics in addition to individual image statistics we can better estimate the correlations between words and visual components.

Acknowledgements

This thesis would not have been possible without the help of my mentors, family and friends.

First and foremost, I express my deepest gratitude to Prof. Lisa Ballesteros for all her time, energy, encouragement and continuous support while supervising me throughout this research project, for being a very inspirational teacher and especially for showing me both the challenges and the fun of doing research. I am also tremendously thankful to Prof. Sami Rollins and Prof. Sanjeev Khudanpur for their guidance and suggestions.

I am also indebted to all the people who took the time to listen to me talking about this project, perhaps not realizing that trying to explain my ideas inevitably set me thinking about those ideas on an entirely new level - Snejina Alexieva, Rosica Dineva, Yanitsa Hristova, Denica Koycheva, Maria Pencheva, and Galina Zapryanova. I am especially grateful to my friends Snejina, Denica and Yanitsa without whom college would have been an entirely uneventful experience.

And last but not least, I thank my parents and sister for always encouraging me in every endeavor, even when it meant going away.

Table of Contents

List of Figures

Chapter 1

Introduction

Nowadays information is abundant and ubiquitous. The foremost example of an information repository is the World Wide Web, which provides everyone connected to the Internet with access to billions of pages. But even though information is available, it is not easy to find. For a person looking for the answer to a specific question, it is very frustrating to know that almost certainly someone has posted it online but to have no idea where to look for it. How to store and retrieve information effectively is an old problem that has been the subject of library science for centuries. However, its scope and focus have changed - today information is largely digitized and often in vast amounts. As a result, there is a compelling demand for computer systems that assist users by discovering, extracting, analyzing and presenting information - systems that act as modern-day librarians in a digital library which spans millions of computers.

Information comes in different forms - text, tables, pictures, diagrams, and sound to name a few, and each format and medium poses different theoretical and practical questions. Images, in particular, are an expressive, universally recognizable source of information which can provide greater detail than written text - as Confucius has said, a picture is worth a thou-

sand words. And in contrast to text, pictures are meaningful to everyone no matter what language a person speaks.

Whether working with images or text, deriving information from a document is not straightforward because we have to decide what to represent and how to represent it. The ambiguity of natural languages makes text documents hard to analyze but the task is even more challenging for images because visual information is inherently more complex. A supplementary feature, which makes analyzing an image easier, is a short piece of descriptive, explanatory text, which we call an annotation.

Annotations are textual representations of images and the focus of this thesis is the process of generating them automatically. More specifically, we develop a method for modeling the relationships between visual features and words that allows a computer system to automatically assign suitable words to previously unannotated images. In the particular context described here, our technique pairs pictures and words. But from a more general perspective it takes one representation of a document and produces another. Therefore, its applicability is not confined to images only. Our method can be adapted to annotating video clips and audio, and in general 'translating' any kind of information from one form to another.

The rest of this thesis is organized as follows: We start by introducing the basic concepts and the main issues involved in the classical problem of searching for relevant information. The second chapter discusses in greater detail fundamental retrieval techniques either directly applied in our work or necessary for analyzing our findings. The third chapter discusses specific techniques for automatically generating image annotations. The rest of the thesis describes our models, experiments and results.

1.1 Information Retrieval

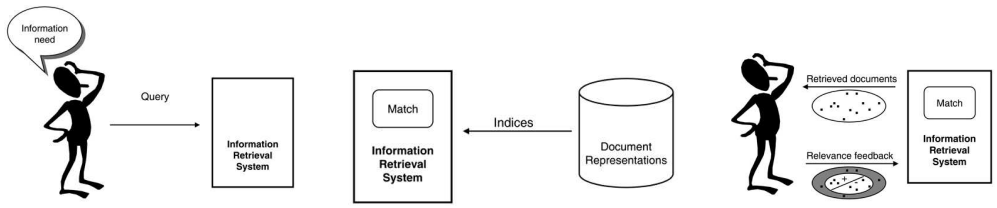
Information Retrieval (IR) is the scientific discipline of automatically indexing, searching and extracting stored information.

As one of the first researchers in the field explains, "[IR] does not inform (i.e. change the knowledge of) the user on the subject of his inquiry, [but] merely informs on the existence (or non-existence) and whereabouts of documents relating to his request".[1] In essence, IR studies and develops techniques that enable computer systems to efficiently store and recover relevant information.

Searching large collections of records in order to find information is a well-studied problem. Library science has developed efficient practices of storing and categorizing written documents to simplify the task of browsing library collections. Building catalogs requires the expertise of librarians to create an index entry for each work in the collection (including factual information such as title, author and publisher, and descriptive information such as summary of contents and major topics) and to structure and organize the index. Reference librarians also serve as intermediaries for library patrons who seek information by providing them with guidance on how to search for resources and how to use the available tools.

Though techniques employed by librarians are very efficient, there is a physical limitation on how much information a person can process and catalog. Nowadays there exist huge electronic collections of digitized materials. For example, the World Wide Web provides access to billions of documents in various formats and languages. The amount of information available on the Internet makes manual management impractical and brings up a need for automatic systems, which locate relevant images in a con-

[1] F. Wilfrid Lancaster. Information Retrieval Systems: Characteristics, Testing and Evaluation, John Wiley & Sons, New York, 1968.



Step 1: The user specifies her information need by typing a sequence of query words.

Step 2: The system searches its database of representations to find documents matching the query.

Step 3: The system returns a list of relevant documents. The user inspects them and provides relevance feedback.

Figure 1.1: The process of retrieving information using an automatic system.

venient and efficient way. Consequently, the continuous expansion of the Web has prompted the emergence of numerous Web-based retrieval systems, popularly known as search engines, such as AltaVista, Google and Dogpile.

One defining characteristic of retrieval systems is that they manipulate unstructured collections of heterogeneous documents, e.g. text excerpts with various lengths or written in different languages or files of different media altogether. Another important aspect of retrieval systems is that they do not answer questions but give pointers to documents which might contain the answer. Queries can be expressed in terms of any combination of words or even formulated as a question, but an IR system only locates documents containing a combination of the query words or their morphological roots. Just like in a library, it is then the responsibility of the user to read through the documents to find the actual answer to her information need.

Retrieval systems differ in design and implementation but the process of finding relevant information always involves three fundamental steps, which are summarized in Figure 1.1. First, the user asks a question: she formulates a query and poses it to the system. This raises many issues

in itself because verbalizing an information need in concrete words is usually not straightforward. During the second step the system searches its database of document representations and recognizes documents containing relevant information. This task implicitly requires interpreting the query as a representation of the user's information need. Thus the information need is first expressed in words and then these words are interpreted to infer the underlying meaning. Neither process is a one-to-one translation as the same question can be expressed in different words, and the actual phrasing can be interpreted in different ways. Finally, the user assesses the quality of the returned subset of documents. Although human-computer interaction, referring to the first and third steps, is an important component of a retrieval system, the focus of this thesis is on the components involved in successfully retrieving information. In short, these are: first representing, organizing and storing large amounts of data; second, determining what an individual document is about; and third, measuring its relevance to a given query.

Although the most extensive research in information retrieval has focused on searching in text corpora, the broad definition of a "document relevant to an information need" encompasses not only text but other media as well, such as pictures, movies and sounds. On the Internet, for example, text in the form of HTML pages, PDF files and links is the predominant source of information, but there are numerous images (JPEG and GIF), video segments (MPEG and AVI) and audio files (MP3) as well. Those documents are inherently more complex in structure than plain text documents, which are organized as a linear combination of strings from a finite vocabulary. As a consequence image, video and audio retrieval differs from text IR and the methodology developed for text IR needs to be either extended or modified in order to manage and access integrated data

collections, which comprise multiple medium types. For this project, we specifically concentrated on image retrieval, a discipline with a broad and diverse set of potential applications.

1.2 Image Retrieval and Its Applications

One promising use of image retrieval is for automatic face detection as part of surveillance systems, where in order to identify a particular person in the crowd, the faces of all those present have to be examined in turn. Clearly, finding shots of the same person implies finding similar images.

Medical imaging is another area where image retrieval could be applied to a great advantage, especially in assisting physicians to make diagnoses by comparing X-ray photos of the patient with that of previous cases. Similar technology could also be helpful in preventing Internet crimes by detecting pornographic images of children, and in protecting intellectual property by identifying illegal versions of copyrighted images.

Image retrieval can also be extremely useful for browsing and auto-illustration. Teachers, for example, could benefit from tools that identify visual materials for exemplifying presentations. Thus, they would spend more time improving their lectures rather than drawing figures. Similarly, journalists could find online images to illustrate their articles and reports, even if they do not have the opportunity to be present at the event themselves.

Another application is organizing retrieval results or image collections, possibly according to user feedback such as the preferences of an individual user. For example, many museums and art galleries are currently working on digitizing parts of their collections and advertising their exhibitions by publishing in the Internet photographs of their most valuable works of art.

Such interfaces will be much more attractive to potential visitors if online exhibits are well-organized and convenient to browse.

The above list of possible applications is by no means complete. The importance and potential benefits of image retrieval are widely recognized and there is a lot of active research in the field, both by academia and industry. Popular search engines like Google and AltaVista provide image-search services and although their capabilities are comprehensive in terms of number of images indexed (Google image search indexes 425 million images), the results are often not satisfactory, sometimes forcing the user to search through many consecutive pages to find the desired image. The algorithm applied by Google relies heavily on metadata such as filenames and captions.[2] Such classification information is implicitly external and while useful in practice it is not sufficient to describe an image in its complexity. The different aspects of representing images in the form of annotations are briefly discussed in the next section.

1.3 Automatic Image Annotation

Text-based image retrieval methods search for images based on associated text, referred to as annotations. These can be created manually but the process is time-demanding and expensive, especially for large collections.

An alternative approach is automatic image annotation, where a computer system relies on a set of manually annotated examples to learn correspondences between words and visual components, and then uses the discovered correlations to annotate new images automatically.

There are two general approaches to automatic image annotation. The

[2]Google's algorithms are propriety and they typically do not reveal much detail about implementation. For a brief explanation how Google Image Search works, visit `http://www.google.com/help/faq_images.html`.

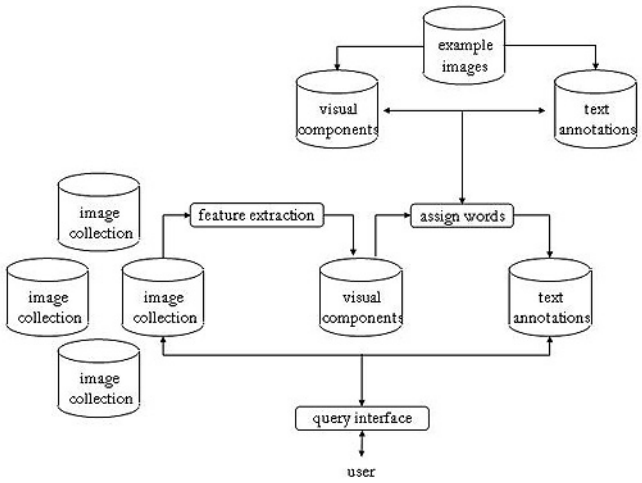


Figure 1.2: General architecture of a text-based image retrieval system including an automatic annotation component.

one used by search engines such as Google is based on information extraction and consists of selecting keywords from surrounding or anchor text. For example, images found on the Web are often used to illustrate a page, so they appear in between some text. Google would analyze the two paragraphs immediately preceding and following an image to create its annotation.[3] The assumption is that text in proximity to the image is its context and therefore contains relevant and descriptive words. In practice, this often does not hold as Web designers might fail to properly describe visual content and advertisers might try to fool search engines and use knowledge of the indexing algorithms to increase their traffic. As a result, annotations are noisy or the search engine misses images altogether.

The other method is assigning words automatically, which can be considered a classification problem. For each word in a controlled vocabulary, an annotation system decides which images to assign to the class corre-

[3]Although this is the general framework of Google's image annotation method, we do not claim to know precisely what portion of the surrounding text they take into account.

sponding to a given word. For example, in our experiments we use annotations that are made up from 1 to 5 nouns and we build a vocabulary of all the words from example annotations. If the images are associated with complete sentences instead, then the annotations can be parsed and tagged for parts of speech, so that only nouns are extracted. The text can be further purged by removing words that occur only a few times, since in such cases there are probably not enough available examples to learn the underlying relationships dependably.

On the visual side of image analysis, low-level visual features such as shape, color and texture are fundamental, but not sufficient, for object recognition. A lesser challenge is for the retrieval system to "understand" the semantics of an image without explicitly recognizing the comprising objects. To achieve this, the system can learn from manually annotated images what correlations exist between words and visual features (visterms), and then use the discovered relationships to automatically assign semantically descriptive words to new, unannotated images. This thesis builds on the Cross-media Relevance model, which learns such correspondences by comparing the visual components of the image to be annotated with those of already annotated images to find word-visterm co-occurrences.

However, it is difficult to measure similarity by considering two images at a time because two images might be easily distinguishable and yet similar with respect to the collection as a whole. For example, if we look at two pictures of a tiger, one showing it lying in the grass and the other - chasing after pray, we would consider them to be different. But if we have a whole collection of wildlife photographs to look at, we would consider the two tiger images to be more similar to one another than to pictures of other animals. This simple example shows how global patterns could bring new information which is not present when analyzing the collection

image by image. Therefore, annotation performance might improve if we take advantage of the hypothesis that similar images are annotated with the same words. Our work extends the Cross-media Relevance model to organize images into groups and thus incorporate the common information shared by similar images.

In Chapters 4 and 6 we describe how to use a clustering algorithm to capture similarities and differences among images across the collection, and then how to use statistics extracted from clusters of similar images in addition to statistics extracted from individual images to better estimate the correlations between words and visual components. But before going into further details, we give a short overview of several fundamental IR terms and techniques.

Chapter 2

Background

Before focusing exclusively on the definition and implementation of methods for automatic image annotation, we first present an overview of the preliminary preparation process involved in retrieving information - document representation. Regardless of what kind of documents a system is designed to retrieve - text, images or another medium, the collection of documents has to be analyzed and represented in a consistent and structured way before the system is actually put in use. This is a kind of summarization, of extracting certain statistical information, which is later used to answer queries quickly and efficiently. For text-based image retrieval, which is based on the annotations of images rather than on the images themselves, generating annotations is part of the representation process.

In this chapter we also discuss clustering - an unsupervised method for partitioning a set into subsets of similar elements. Clustering is particularly relevant to our work since, as already mentioned, we are going to use groups of similar images to improve annotation and hence retrieval.

2.1 Document Representation

The first task of a retrieval system is to index the document collection so that searching is fast and efficient. Internal representations of documents are much simplified. The system cannot store the full text because of storage concerns, but this is not necessary either, since not all the words in a document are good descriptors of its content. Therefore, we have to decide what information is important and should be preserved, and what information could be discarded.

Depending on how much information about documents is preserved, representations have different *granularity*. Here is a list of different levels of granularity, ordered from the coarsest to the finest:

- We only keep track of whether a word appears in a document or not.
- We keep track of how many times a word appears in a document.
- We keep track of all the positions at which a word appears in a document.

Obviously, what granularity we choose makes a difference both in terms of storage space and in terms of average search time and search options that can be supported, because it takes longer to go through a list of occurrences than to check a single number. On the other hand, the more information we choose to index, the more elaborate our analysis of relevance is. For example, if we keep track of positions of occurrence, we can define phrases as two words which appear together within a window of a certain length, and query on phrases as well as individual words.

The number of times a word appears in a document is referred to as its Term Frequency (TF). TF attempts to measure what the document is about. As the author of an article would naturally use specific concepts and

terms to develop her ideas, words that are pertinent to the main themes under discussion would turn up more often than general, loosely related words. Therefore, words occurring frequently in a document will convey more about its topic than rarely occurring words.

On the other hand, some words are more frequent than others in the collection as a whole. If a word appears in a great number of documents, it is likely to appear in both relevant and nonrelevant documents. Recall that the retrieval system does not attempt to understand the content of a document but only to judge documents as relevant or not relevant to a specific information need. Because common words fail to contribute to the analysis of relevance (they have less information content), the system should give less importance to the occurrences of such words. In fact, some words appear so often (e.g. 'the', 'a', 'of', 'to', 'and', 'in') that they are often discarded altogether - these are called *stop words*. Since the few hundred most frequent words account for about 50% of natural language text, there is a doublefold advantage in removing stop words - we significantly reduce the necessary storage space and we dispose of words that have little semantic importance.

The overall number of documents in which a word occurs is referred to as its Document Frequency (DF) and the value $log(N/DF)$, where N is the size of the document collection, is referred to as its Inverse Document Frequency (IDF). IDF attempts to measure a word's power to discriminate between relevant and nonrelevant documents. Words such as prepositions, pronouns and the verbs 'say' and 'become', are so common that they are hardly useful for retrieval, since queries tend to be specific. On the other hand, if a query word is relatively rare, then we could be fairly sure that documents containing it are relevant. IDF factors in this reasoning by deemphasizing words that appear often across the collection.

To combine these two measures, most retrieval systems weight terms by their TF×IDF values.

When counting occurrences, we should also consider document length. Clearly, there is a higher probability for a word to appear in a long document just because it consists of many words. Therefore, it is reasonable to assume that a certain number of occurrences in a short document are a more significant indicator of relevance than the same number of occurrences in a longer document. (We can think about this as the word being more central to the major topics discussed in the first document.) Without document length normalization, words in long documents can be perceived as more important due to more occurrences and higher TF weights.

One issue that cannot be solved simply by counting occurrences is *synonymy*, the fact that several words might mean the same thing, yet the user types only one of those. A traditional IR approach for addressing the notion of synonymy is *query expansion*. It is implemented by performing an initial query and then augmenting it with salient words from the top documents. Query expansion improves performance because it captures the context in which a word appears and this includes not only synonyms specifically, but concepts associated with the same topic in general.

Since our particular task is image annotation, it might seem at first that the issue of synonymy has no relevance in a non-linguistic setting. However, as we will describe in Chapter 4, our approach considers annotation to be a retrieval problem where we rank candidate words in terms of their likelihood to be related to the image. The same word can be used to describe very different image samples, e.g. 'water' can be associated with all shades of green and blue - from an IR point of view we have different visual features that mean the same thing. Therefore, how to deal with synonymy is still an important question in the context of image annotation.

2.1.1 Inverted Index

Counting occurrences and computing TF×IDF weights for documents can be done offline, so that at search time results the necessary processing is completed quickly. The data structure used for this purpose is called an *inverted index*. It is a representation that consists of a lexicon and a collection of postings. The lexicon is a list of distinct terms occurring in the collection, except for removed stop words. The entries are ordered alphabetically for faster searching and each entry keeps some statistics about the corresponding term, such as IDF, which will be used in ranking computations. A posting is a list of the documents which contain a given term, the number of occurrences in each document and possibly the exact text positions where the term occurs. Lexicon entries hold a pointer to the posting of their corresponding term, which is also referenced when computing the degree of similarity between a document and a query.

The inverted index is constructed to speed up retrieval - in order to find where a particular word occurs and with what frequency, it is first looked up in the lexicon which points directly to the posting containing this information. But while the lexicon does not take up much storage space, the postings require large amount of space, particularly if recording more detailed information such as text position - in this case, each word in the text would be referenced once in the structure (again, excluding stop words).

2.2 Similarity Measures

A similarity measure is a function which takes two objects of the same kind and returns a measure of how similar they are or, alternatively, of the distance between them. The concept of distance is meaningful only

for objects which are represented in a metric space, while similarity is a more general concept. Intuitively, distance and similarity are inversely proportional: the greater the similarity between two objects, the smaller the distance which separates them, and vice versa. A similarity function is almost always symmetric, i.e. $distance(a, b) = distance(b, a)$. Example of a non-symmetric distance measure is the Kullback-Leibler divergence, which measures the difference between two probability distributions and is often used in information theory.

The vector space representation assumes that terms are independent and disregards positional information specifying how words are arranged in meaningful sentences (hence, phrases which are combinations of consecutive words, cannot be straightforwardly represented). The advantage is that the similarity between a document and a query can be easily formulated and computed once both the documents and the query have been represented as a vector in V-dimensional space, where V is the size of the vocabulary. By measuring the of degree similarity we quantify the relationship between a document $D = (t_{1,D}, t_{2,D}, ..., t_{V,D})$ and a query $Q = (t_{1,Q}, t_{2,Q}, ..., t_{V,Q})$, which then allows us to estimate how relevant the document is to the query.

One similarity measure widely used in text retrieval is the cosine of the angle formed by the two vectors:

$$\begin{aligned} sim(D, Q) &= \frac{D \circ Q}{\mid D \mid \times \mid Q \mid} \\ &= \frac{\sum_{i=1}^{V} t_{i,D} \times t_{i,Q}}{\sqrt{\sum_{i=1}^{V} t_{i,D}^2} \times \sqrt{\sum_{j=1}^{V} t_{j,Q}^2}} \end{aligned}$$

The factor $\mid Q \mid$ does not affect ranking as it the same for all documents. The other multiple in the denominator $\mid D \mid$ plays the role of document length normalization.

An alternative measure is Euclidean distance which measures dissimilarity instead of similarity, although similarity and dissimilarity can be normalized to add up to 1.

$$dist(D, Q) = \sqrt{\sum_{i=1}^{V} (t_{i,D} - t_{i,Q})}$$

2.3 Evaluation Measures

The evaluation of a computer system designed to interact directly with users is never straightforward or easy due to human factors. This is particularly true for information retrieval systems because the notion of relevance is subjective, as people might interpret differently the meaning of a document depending on presumed information need or state of current knowledge.

Therefore, it is important to acknowledge that human participation is an essential part of an IR system. Although features such as information visualization and user interfaces should not be ignored by developers of search engines, there are inherent complexities involved in working with humans because people with different backgrounds, experiences and motivations would perceive relevance and therefore search results in different ways. To stay away from these complexities, researchers have devised precise evaluation methods based on having a reference test collection. Such a collection consists of a set of documents, a set of example queries and relevance judgments which specify the subset of relevant documents for each query. The relevance judgments are provided by humans and in this aspect it can be argued how objective they are, but this method allows at least straightforward comparison between different retrieval systems.

The two most widely used evaluation measures are *recall* and *precision*. They are used to estimate the performance of a retrieval system on a single query in terms of the number of retrieved documents $Retr$, which are returned by the system, and the number of relevant documents Rel, which are known to contain pertinent information.

2.3.1 Recall

Recall measures the ability of the system to find relevant information. It is defined as the fraction of relevant documents which have been effectively retrieved:

$$Recall = \frac{\mid Retr \cap Rel \mid}{\mid Rel \mid}$$

High recall means that there might be many nonrelevant documents in the retrieved set, but most of the relevant ones are also included.

2.3.2 Precision

Precision measures the ability of the system to ignore nonrelevant information. It is defined as the fraction of retrieved documents which are considered to be relevant:

$$Precision = \frac{\mid Retr \cap Rel \mid}{\mid Retr \mid}$$

High precision means that many relevant documents might not be retrieved at all, but the set of retrieved documents mostly consists of relevant ones.

There is an implicit tradeoff between recall and precision. Even without improving the ranking, we can increase recall simply by retrieving more documents. At the same time, however, this will significantly hurt precision

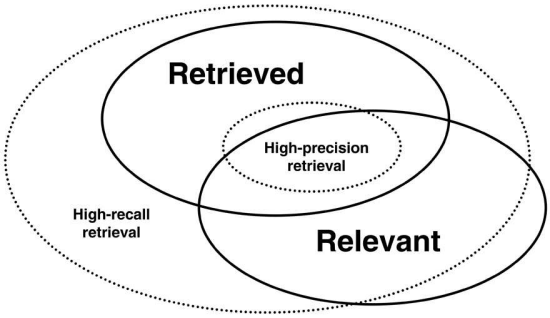


Figure 2.1: Comparing the set of relevant and the set of retrieved documents.

as we may retrieve more nonrelevant documents as well. Similarly, we can increase precision by retrieving only a small subset of documents with high confidence of being relevant but this will significantly hurt recall as we leave out many other good documents. Figure 2.1 graphically shows this relationship. In practice, there are situations in which the user would prefer one of those scenarios to the other. For example, a person using an online search engine will probably look only at the first 10 to 20 documents. On the other hand, a lawyer trying to find precedents will probably want to go through every document even slightly connected with her case. In the former scenario the user wants high precision, in the latter - high recall. Such considerations underscore how important the human factor is for the effectiveness of a retrieval system and they should be taken into account during the design process.

2.3.3 Mean Average Precision

Recall and precision evaluate performance as the intersection of two sets assuming that a document is either relevant or nonrelevant. Intuitively, this assumption is false. Since an information need is inherently imprecise, the query generated to express it is inherently ambiguous. Consequently,

relevance is inherently not boolean and can take any real value between 0 and 1. Therefore, in most retrieval systems the documents retrieved in response to a query are ranked according to their inferred relevance to the query.

The performance metric used to evaluate ranked retrieval is called *average precision*. It combines recall and precision by averaging precisions at different recall points. Starting from the top document in the list and keeping track of the subset of documents retrieved so far, we compute precision whenever a relevant document is encountered in the ranked list and therefore recall increases. Mean average precision is the arithmetic mean of average precision for a set of queries. The precise mathematical definitions for average precision AP, given a query q, and mean average precision mAP, given a set of queries Q, are given below:

$$\begin{aligned} AP(q) &= \frac{\sum_{r \in Rel(q)} P(r)}{\mid Rel(q) \mid} \\ mAP &= \frac{\sum_{q \in Q} AP(q)}{\mid Q \mid} \end{aligned}$$

where $P(r)$ is precision at rank r and the document at r is relevant. Average precision is visualized by drawing recall-precision curves which show precision at standard recall points, e.g. 10%, 20%,..., 100%, and interpolate between points.

The same definitions of recall, precision and mean average precision can be used to measure the performance of an image retrieval system. We can evaluate a model on a portion of the annotation examples, which have been put aside for testing and not used for estimating model parameters. The manual annotations are assumed to be the ground truth and ideally the system would reproduce all and only the manual annotations. Thus,

in order to measure annotation performance for a given word we compare the generated and the manual annotations of test images. The relevant documents are those images whose manual annotations contain the word; the retrieved documents are those images whose generated annotations contain the word.

2.4 Vector Space Model

A simple Boolean retrieval model represents documents as binary vectors in V-dimensional space, where V is the number of distinct terms in the collection. The components of a document vector are set to either 1 or 0 depending on whether the corresponding index term is present or absent in the document. Although this framework provides enough information to make a binary decision (the document is either relevant or nonrelevant depending on whether it contains the query word or not), it is insufficient to measure the degree of similarity between a document and a query and therefore to rank documents with respect to their relevance. The classical Vector Space retrieval model makes partial matching between a query and a document possible by assigning non-binary weights to the index terms, e.g. computed using TF×IDF weighting. Given a query, the model estimates for each document a value which is interpreted as a measure of how well the document matches the query and therefore of how much it is relevant. A straightforward result of computing degrees of relevance is the capability to rank documents in terms of how well they are supposed to satisfy the information need.

2.5 Latent Semantic Indexing

One IR technique for analyzing the context of terms more extensively is Latent Semantic Indexing (LSI). It is based on a straightforward mathematical procedure for reducing the rank of a matrix and was first proposed by Deerwester *et al* [7].

TF and IDF are computed independently for each term. Consequently, these measures fail to capture how terms are related across the collection. However, if two terms tend to appear in the same documents, they should be considered interconnected. LSI attempts to address this problem by mapping related terms to a single term.

If we have represented documents as vectors in V-dimensional space, we can construct an index matrix M with documents along the rows and terms along the columns. Then component M_{ij} reflects the weight of term j for document i. In the worst case, this matrix has rank $R_M = V$ although it is possible that $R_M < V$ if there are linearly dependent terms.

LSI analyzes co-occurrence patterns existing in the document-by-term index matrix to discover semantic relatedness among terms. If two (or more) terms tend to occur in the same documents, they are projected onto the same dimension in the latent semantic space, which implies that they are mapped to the same concept. Thus, even if a document does not match any of the query words, it can still be considered relevant if it is similar to the query in the concept space created by LSI.

2.5.1 Singular Value Decomposition

LSI uses Singular Value Decomposition (SVD) to project documents and queries into a low-dimensional space. According to the following linear algebra theorem,

For every $m \times n$ matrix $\boldsymbol{M} : R^n \rightarrow R^m$, there exists a factorization in the form $\boldsymbol{M} = \boldsymbol{U}_{[m \times n]} \boldsymbol{S}_{[n \times n]} \boldsymbol{V}_{[n \times n]}$.

Here $\mathbf{S}$ is a diagonal matrix: all its entries are zeros except for those along the main diagonal. Moreover, both $\mathbf{U}$ and $\mathbf{V}$ are orthogonal, so that $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ and $\mathbf{V}^T\mathbf{V} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. The components along the main diagonal of $\mathbf{S}$ are the singular values or *eigenvalues* of $\mathbf{M}$. Taking a subset of the singular values and the corresponding columns of $\mathbf{U}$ and the corresponding rows of $\mathbf{V}$ gives us a low-rank approximation of $\mathbf{M}$:

$$\mathbf{M}' = \mathbf{U}'_{[m \times k]} \mathbf{S}'_{[k \times k]} \mathbf{U}'_{[k \times n]}$$

Of course, by ignoring singular values we discard some information, but taking the k biggest singular values guarantees that we get the closest approximation for the particular value of k (closest in terms of squared error since the approximation error is determined by the sum of the squared singular values that were excluded).

This transformation maps $\mathbf{M}$ into a low-dimensional space. An obvious benefit is saving space because the system needs to store a much smaller matrix and a bigger part of it will fit into main memory. However, this strategy has an additional, subtler advantage: it attempts to automatically find associations between related terms, the very analysis which makes people so good at judging relevance. If a retrieval system is unable to identify how terms are related, it can only find documents which directly match the query by containing all query words. However, there are words with the same meaning, and by not taking such words into account the system may miss relevant documents (the issue of *synonymy*). Also, words usually have several meanings and the system might retrieve documents which are actually nonrelevant (the issue of *polysemy*). One approach for dealing with synonymy and polysemy is using SVD and a low-rank approximation

of the index matrix. (There are others as well, such as query expansion and a thesaurus, but we are not looking into these here.)

An important point to make is that LSI has no underlying linguistic foundation. Rank reduction and matrix transformations do not perform linguistic analysis of the meaning of text; they are mathematical tools for analyzing co-occurrence patterns. A system which uses a different technique to investigate co-occurrences may be able to identify the same or better word relationships.

Deerwester *et al* show that LSI performs better than classical vector space retrieval, especially for high-recall searches. LSI increases recall because it finds relevant documents that would be overlooked in case of straightforward term matching. But at the same time, LSI can hurt precision by assessing as relevant documents which are in fact nonrelevant and by reducing very fine differences to the same level of detail.

The LSI approximation of the index matrix is $\mathbf{U}'\mathbf{S}'\mathbf{V}'$, where the columns of $\mathbf{U}'$ and the rows of $\mathbf{V}'$ are in a concept space of size k. Thus, after the transformation documents are represented in terms of k concepts rather than n terms and by using $\mathbf{U}'$ and $\mathbf{V}'$ a retrieval system can retrieve on concepts instead of terms. The document and term mappings are $\mathbf{U}'$ and $\mathbf{V}'$, respectively. In essence, LSI considers documents which have many terms in common to be semantically similar and documents which have few terms in common to be semantically unrelated.

After LSI transformation the coordinates of document vectors no longer explicitly reflect term frequency. In fact, LSI data is not easily interpreted as it has no direct semantic meaning and the generated concepts are not directly related to what a human would consider a concept. For example, LSI does not explicitly find groups of words that all refer to the same object or idea: while 'Bengal' and 'tiger' are not synonyms in the literal sense, LSI

might consider those words to be the same concept as they often appear together. By decomposing the index matrix the retrieval system does not begin to understand what words and documents mean, but takes advantage of global word usage patterns to find related terms.

Latent semantic indexing reduces the rank of the index matrix to k. While there are many possible k-rank approximations, LSI guarantees that we use the optimal one in terms of squared error, and therefore the one which incurs the smallest loss of information. However, choosing the optimal value of k is an open question. This is usually done empirically rather than taking into consideration the characteristics of the collection. The inherent difficulty lies in the fact that LSI tries to model hidden relationships and it is not clear how obvious features such as the size of collection and the average size of documents are related to the underlying the semantic structure of the collection. Another potential problem is the computational cost of performing singular value decomposition on a huge matrix. SVD has complexity $O(n^2k^3)$ where n is the number of terms plus the number of documents and k is the rank of the approximation. Although k is kept small (in the range 100 to 300), n grows fast as the size of the collection increases. However, the computational cost involved might be acceptable since LSI needs to be performed only once for a static collection.

2.6 Language Modeling

Language modeling (LM) is a probabilistic method that has long been used in speech recognition, optical character recognition and machine translation. In all these applications, LM is used to choose between several alternative hypotheses, which are considered highly likely. For example, a speech recognizer might be unable to distinguish between 'trait' and 'trade'

given the acoustics alone. In order to disambiguate, the system can use a general model of the English language. Let assume that the next word is 'union'. Since 'trade union' is more likely to appear in an English sentence than 'trait union' , the recognizer should choose 'trade'. This is an example of a bigram language model, one that considers two instances - the current and the previous observation.

LM is also used in information retrieval. With this approach the system estimates a language model from each document in the collection. (Internally, the representation of a document is a probability distribution rather than a multidimensional vector.) Then, the models are used at search time to rank documents in terms of the probability that the query is generated by the corresponding model.

A language model is built by counting the word occurrences in the document and assigning probabilities to words - the more frequently a word appears, the higher probability it gets since it is considered more descriptive of semantic content. Notice that LM explicitly looks only at the term frequency TF of a word. This could suggest that the Vector Space model is superior because it also takes into account the document frequency through TF×IDF weighting. In fact, it can be shown that LM includes an IDF component in its estimations as well, though not explicitly. We return to this discussion in Section 4.1.1.

2.6.1 Relevance Modeling

Relevance modeling (RM) is a generative Language modeling (LM) approach but while the classical LM considers documents to generate queries, RM assumes that both the documents and the query are generated by an underlying conceptual model R, which describes a particular information need. This is an accurate representation of reality since it is possible to

express an information need in various ways, for example by using words and phrases that have similar meaning. Therefore, one and the same relevance model R can generate different queries and several documents that are relevant to those queries.

RM introduces query expansion, which is discussed in Section 2.1, in a language modeling setting. The process of query generation is directly modeled at search time (in contrast to other methods who precompute probability tables) and the algorithm ranks documents on the probability of observing the query terms during a random sampling from the underlying probability distribution. As with other query expansion techniques, there is an initial retrieval step but rather than explicitly selecting additional terms, RM compares the query with the top documents, re-estimates probabilities, and then re-ranks documents.

RM has a formal theoretical foundation and yet a simple, intuitive and well-understood framework. It does not rely on specific knowledge about the syntactical rules of the language in which the documents are written; hence it is straightforward to apply in cross-language retrieval or in monolingual retrieval in languages for which no linguistic theory has been developed. This last characteristic makes the RM especially suitable for annotating images (we consider this to be a cross-media retrieval task, where the two media are image and text), because the language of visterms is linguistically unstructured. In this work, we extend the original definition of relevance modeling by including information extracted from groups of similar documents in addition to information extracted from individual documents.

2.7 Image Retrieval Techniques

Image retrieval techniques are classified into two types, text-based (TBIR) and content-based (CBIR), and the foremost difference between those is the internal image representation. While text-based methods represent images indirectly in terms of annotation, caption, file name and/or surrounding text, content-based methods represent images directly in terms of their visual features.

Representation determines how images are indexed and searched, and consequently how queries are formulated. Text-based image retrieval allows for a query-by-text approach where the query is expressed in words and the system should return images whose annotations are most relevant, e.g. the user asks 'tiger in a natural environment' and gets back images of tigers in the savanna or in the jungle. Content-based image retrieval allows for a query-by-example approach where the query itself is an image and the system should return images whose visual features are most similar to those of the provided image.

Because of the differences outlined above, the two types of image retrieval techniques are suitable for different application domains. Text-based retrieval is practical in a more general setting, e.g. browsing a collection of digital photographs where images are automatically arranged based on keywords extracted from their annotations [19], while content-based retrieval is practical in a more specialized context, e.g. face detection for surveillance purposes [22].

Although querying by example and hence content-based image retrieval have many potential applications, querying by text is actually more convenient for users. An Internet user would most probably not search for sunset images if she already has one. So she would have to first draw a picture resembling a sun disappearing behind the horizon, e.g. an orange-

to-red circle on a blue background. This seems relatively easy but imagine drawing a human face or a tiger! Because content-based techniques require thinking (and visualizing) what we are searching for in terms of low-level visual features such as color, shape and texture, they are unintuitive and unnatural. Moreover, tigers are taken picture of from different angles and therefore appear in different shapes and colors. So even if the user successfully draws a tiger turned to the left, the system will be able to find only very similar tigers while the user probably wants to search for any tiger regardless of the way it faces the viewer.

Because querying by providing an example is cumbersome, using a verbal description to specify the desired images and hence text-based image retrieval are advantageous in interactive search systems. Users rely on textual information more than on visual information to validate their search results, as demonstrated by a study on the role of text and image representations of video segments reported in [10]. In this investigation, the retrieval system displayed both textual (title and description) and visual (three frames) surrogates for the video information and an eyetracking device was used to determine which representation users looked at first and more often. Most of the subjects used the text to make their relevance judgments and the images afterwards to confirm the selection. A similar conclusion is made in another study on the utilization of textual and non-textual relevance criteria for judging the relevance of photographs [6].

Text-based image retrieval represents images in terms of keywords or concepts. Therefore, systems can search for relevant images based on semantics rather than appearance and users can formulate queries in a natural language. However, the process of acquiring the annotations necessary for building such a system is not trivial. Annotations can be assigned manually but to have someone work through the entire collection, analyzing every

single image, would be time-consuming and expensive since it would require training annotators, the way reference librarians are trained to work with paper documents and archives. In addition, human annotations are inherently subjective as they reflect the annotator's personal view and understanding of the image. Human annotation relies on the person to decide what the most salient objects are. Common, background objects such as the sky might be left out because they seem trivial to the annotator. Therefore, images might be labeled inconsistently when several people work on the same collection.[1]

The annotation model proposed in our work attempts to improve the cross-media relevance modeling approach by incorporating clustering. The idea is motivated by the fact that human annotations are often incomplete: sometimes an object that is obviously in the picture doesn't appear in the annotation. On the other hand, visterms are imperfect descriptors of images as the same visterms can appear in semantically unrelated images. The hypothesis is that a cluster-based Cross-media Relevance model will deal better with these difficulties than the classical Relevance model since it will use the global similarity between images and thus capture correspondences between words and visterms more accurately.

2.8 Clustering

Clustering is an extensively studied technique with various applications in information retrieval, including organizing and browsing collections and retrieval results, interactive relevance feedback and summarization. Clus-

[1]Of course, the annotations used to train an automatic system for generating annotations are still produced manually and therefore are susceptible to human subjectivity. However, since we need only a small portion of the collection for training, the consistency problem could be efficiently dealt with. Also, more time could be allocated per image, so that the manual annotations are carefully created and validated.

tering attempts to utilize similarity relationships between documents. In the Vector Space retrieval model a document is represented as a vector in multidimensional space, where each dimension corresponds to a term in the vocabulary and the terms are weighted by their importance as indicated by the number of occurrences in the given document and across the collection. Queries are represented in the same way. The degree of similarity between two documents (or between a document and a query) is computed on the basis of how many terms they have in common, e.g. using the inner product of two vectors (Euclidean distance) or the cosine of the angle they determine (cosine coefficient). Similar documents would be represented by vectors that are close to each other in space. Therefore, the distance between the index vectors is inversely related to the similarity of the corresponding documents.

The cluster hypothesis, proposed by C. J. van Rijsbergen, states that "closely associated documents tend to be relevant to the same request".[2] It implies that similar documents are about the same topic and therefore groups of similar documents should be retrieved together. Ideally, documents which are relevant to the same information need are clustered together and can be retrieved in response to the same queries. Likewise, documents which are far apart in the document space are not expected to appear in the same context. Thus clustering can increase performance by bringing relevant documents together and separating nonrelevant ones. If we know that a certain subset of documents are relevant to the query and we also know that these documents are similar to another subset of documents, then we conclude that we should combine the two subsets and retrieve them together. Therefore, we could consider clustering to be a learning algorithm that allows us to analyze relationships between docu-

[2]C. J. van Rijsbergen, *Information Retrieval*, Butterworths, London, 1979.

ments and make inferences and generalizations.

Another classifying procedure used in information retrieval is *categorization*, where a hierarchy of categories is chosen manually in advance and a document is assigned to one category or another depending on its characteristics. However, to define a general set of categories is a difficult task, as it requires deciding what categories are most applicable and how they are related. In contrast, clustering is unsupervised and the groups are discovered automatically. And since there are no category labels provided, clusters have no explicit meaning: though we know that documents grouped together are similar in some respect, we do not know the central theme that connects them.

Again, we emphasize the fact that when using clustering we make the assumption that documents in the same cluster have a topic in common. (If this premise is true, then it is reasonable to conclude that we should retrieve all these documents whenever the query refers to the underlying topic.) In practice, documents are clustered in terms of their vector representations and therefore on the basis of purely mathematical properties: clusters combine documents whose corresponding vectors are close in vector space. However, there is no linguistic theory which guarantees that geometric adjacency stems from semantic similarity. In this respect, clustering makes a very strong assumption.

We paraphrase the cluster hypothesis to say that "similar images tend to be annotated with the same words" and we investigate whether this statement is true by using statistical information, shared by images clustered together, to extend the original Cross-media Relevance model. And because textual and visual information complement each other, training images are partitioned into groups based both on their visual characteristics and annotations.

2.8.1 Static & Query-Specific Clustering

Clustering is either *static*, performed prior to answering queries and over the entire collection, or *query-specific*, performed at search time on the individual documents retrieved in response to a given query. Though some studies demonstrate the benefits of using static clustering [21], there is no consistent evidence proving that it effectively improves retrieval performance for general collections. This could be explained by the fact that general relationships do not conform well to specific information needs. A query defines its local context and the general structure of the collection might not be pertinent to individual queries. Query-specific clustering techniques are more flexible because they adapt better to local context [9].

Clusters can also be updated according to local feedback, a retrieval technique which iteratively refines search results in response to relevance judgments provided by the user on the previous set of retrieved documents. The strategy can also be implemented automatically: the system retrieves once using default model parameters, assumes the highest ranked documents are relevant and re-estimates term weights or expands the query based on the initial result. This approach for generating relevance feedback is based on the following observation. Most users inspect only the first couple of pages of retrieved documents, which amounts to the 20 to 30 highest ranked documents. Even if the search engine finds thousands of relevant documents, it is highly unlikely that someone would take the time to look through all of them. Let us assume that if the user clicks on the link to a retrieved page, she makes a positive relevance judgment for this document. We can generalize this line of reasoning by making the further assumption that the user always inspects the first N documents. In this case, the system does not need someone to actually sit down and click on the links - it can directly consider the top N documents to be relevant.

Whether manually or automatically generated, feedback information can be used to recluster documents and find clusters which are better structured with respect to the individual query.

Separating dissimilar documents increases precision as the system avoids retrieving nonrelevant documents since they are far away; bringing together similar documents increases recall as the system retrieves more relevant documents since they are grouped together. Therefore, clustering helps retrieval only if it finds well-formed (tight and clearly separated) clusters. However, clustering itself does not transform document representations - performance depends on the initial configuration of the document space. If term vectors are not well separated, a clustering algorithm will fail to find good clusters. Deciding how to represent documents is therefore crucial: this determines distances between individual documents and, consequently, how successfully a clustering algorithm can minimize intracluster distances and maximize intercluster distances (in order to separate unrelated documents and bring together related ones). Recall from Section 2.1 that term selection and weighting can influence performance. For example, if we use only TF to represent documents we will measure the informativeness of words within documents but not across documents, which is not enough to get a clear separation between relevant and nonrelevant documents.

A clustering technique can be defined in terms of an abstract similarity measure and then implemented in terms of a particular metric, e.g. Euclidean distance or cosine coefficient. Clustering techniques are classified into two major groups depending on the structure of the partitioning they generate. Hierarchical clustering produces a hierarchical tree of clusters where each node unites a pair of related subclusters. Figure 2.2 shows an example of such a hierarchy, called a *dendrogram*. The leaves of the den-

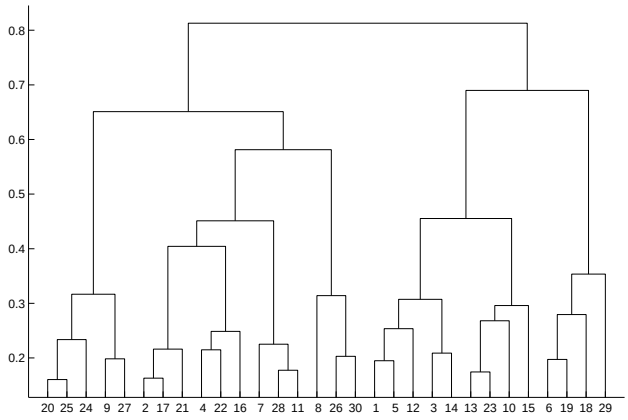


Figure 2.2: Dendrogram representation of hierarchical clustering: At each internal level a horizontal line segment merges the two intermediate clusters with greatest similarity.

drogram are individual documents and their relative similarity is indicated by the height of the node that joins them together. Thus, lower internal nodes represent smaller clusters of closely related documents and nodes higher in the tree represent larger clusters of loosely related documents. Flat or non-hierarchical clustering simply assigns objects to one of a given number of clusters without deriving relationships between clusters.

The choice of which clustering technique to apply should depend on the features of the items to be clustered and the desired characteristics of the clusters themselves [17]. Next, we describe several specific clustering algorithms.

2.8.2 Agglomerative Clustering

Agglomerative clustering is a type of hierarchical clustering which creates the dendrogram bottom-up. An algorithm of this kind starts with each element of the set in its own cluster and then iteratively combines the two closest clusters to form larger ones until there is a single cluster left. Or alternatively, the process stops when the desired number of clusters is

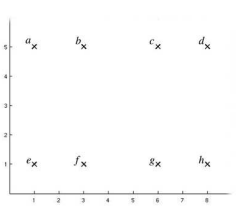
reached.

Agglomerative clustering computes a distance matrix which specifies the distances between objects to be clustered. However, after the two closest objects are merged, the distance matrix is no longer accurate - first because two of the clusters represented no longer exist as separate entities, and second because there is a new cluster whose distances to each of the other clusters are unknown. While removing the two rows and columns in the matrix which correspond to the merged clusters is trivial, to deal with the second issue we need to define a similarity function that will determine how the new distances are computed. Within the general framework of a bottom-up clustering procedure, there are several versions of agglomerative clustering which differ only in their similarity function.

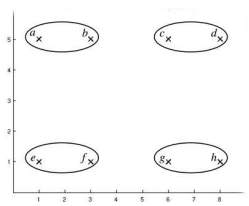
Single-Link Clustering

Single-linkage defines the distance between two clusters be the distance between the two most similar members. So, to update the distance matrix, we need to compare all pairs of objects from two different clusters and take the smallest distance (or greatest similarity). Because the similarity function looks at adjacent objects, it is locally defined and therefore it produces clusters with good *local coherence*. This means that objects that are close to each other are grouped together but it also leads to a tendency to form long, 'straggly' clusters. An example of a straggly cluster is presented in Figure 2.3.

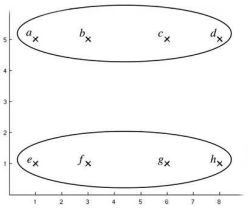
In retrieval, clustering is used to analyze the structure of a document collection and the discovered similarities are used to generalize about overall occurrence patterns. Therefore, producing loose clusters, even if they have good local coherence, could be considered an undesired property because we are interested in creating clusters with good global qualities, i.e. clusters



The initial configuration with eight points to be clustered: a, b, c, d, e, f, g, h.



Following initial four merges, there are four small clusters: $\{a, b\}, \{c, d\}, \{e, f\}, \{g, h\}$.



After two more merges, there are two straggly clusters: $\{a, b, c, d\}, \{e, f, g, h\}$.

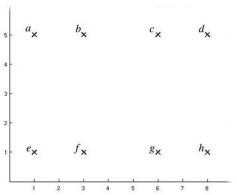
Figure 2.3: This example shows how single-linkage clustering can produce long and loose clusters. Notice how in the final configuration a and d are grouped together while a and e are assigned to different clusters, although a is closer to e than to d.

that are compact with respect to the rest of the clusters.

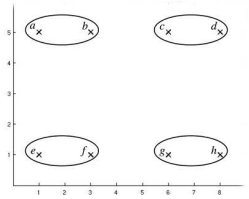
Complete-Link Clustering

Complete-linkage defines the distance between two clusters to be the distance between the two most dissimilar members. As with single-linkage, to update the distance matrix we need to compare all pairs of objects from two different clusters but in this case we take the greatest distance (or least similarity). Because this similarity function is globally defined, it avoids chaining objects in elongated clusters and produces tight spherical ones. An example of a spherical cluster is presented in Figure 2.4.

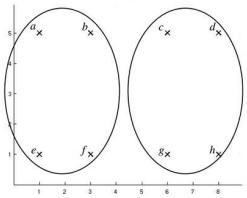
Although theoretically single-linkage might be the "correct" clustering algorithm to use in specific situations, generally we would prefer tighter clusters because they would not contain elements that are strikingly dissimilar. (This could happen within a very long cluster.) In our particular situation we will use clustering to find groups of images assumed to be similar in some aspect: all pairs of images within a cluster should share a commonality, a unifying feature or theme. We could imagine this common feature as a focal point around which cluster members are centered (and



The initial configuration with eight points to be clustered: a, b, c, d, e, f, g, h.



Following initial four merges, there are four small clusters: $\{a, b\}, \{c, d\}, \{e, f\}, \{g, h\}$.



After two more merges, there are two spherical clusters: $\{a, b, e, f\}, \{c, d, g, h\}$.

Figure 2.4: The same example as in Figure 2.3 using complete-linkage clustering. The first four merges are the same but on the next step a and e are clustered together. The final result are two spherical and more compact clusters.

none of the elements should be too far away from this central point). Thinking about this idea geometrically, we could see how it would correspond to spherical rather than elongated clusters.

For a collection of n documents agglomerative clustering completes in $n - 1$ steps. Computing the initial similarity matrix of n documents requires calculating distances between all pairs of documents in the collection, which amounts to $\frac{n(n-1)}{2}$ computations. Each merging step of single-link clustering requires updating the matrix by comparing the respective distances from the two merged clusters A and B to each of the other clusters C:

$$distance(A + B, C) = min\{distance(A, C), distance(B, C)\}$$

Each comparison can be done in constant time as we already know the distances between individual objects but we need to make $n - k - 1$ comparisons where k is the current dendrogram level. Thus, single-linkage

takes at most $\frac{n(n-1)}{2} + (n-1)n = \frac{3}{2}n^2$ steps, so it has overall complexity of $O(n^2)$.

Complete-linkage is more computationally expensive because each iteration of the merging algorithm requires $O(n^2)$ comparisons to find the greatest distance between any two elements for each pair of clusters. Again, there are $n-1$ merging steps for an overall complexity $O(n^3)$.

Group-Average Clustering

Group-average defines the distance between two clusters as the average distance between all pairs of members (including pairs originally from the same cluster). If this computation is not optimized each update of the distance matrix will take at most $O(n^2)$ steps. However, if the objects to be clustered are represented as length-normalized vectors, the update can be completed in linear time; in this case group-average clustering partitions a set of n elements in $O(n^2)$ steps. Thus, it is a compromise between single and complete linkage because it is less computationally intensive than complete-linkage but it also has less tendency to produce straggly clusters.[3]

2.8.3 Non-Hierarchical Clustering

Non-hierarchical clustering techniques take the number of clusters as an input argument and refine an initial partition of the element set iteratively until there is improvement of cluster quality.

The most widely used non-hierarchical algorithm is K-means [17]. It requires specifying the desired number of clusters explicitly. Then it partitions the element set into hard, non-overlapping groups. Given a set of

[3]Refer to [17] for a derivation of $O(n^2)$ running-time complexity for average-linkage in the case of vector-space representation.

n elements to be clustered into k disjoint subsets, the procedure starts by randomly selecting k elements as the initial centroids. A *centroid* is the weighted average of the elements comprising a cluster. By computing the distances from an element to each of the centroids the algorithm finds the closest centroid and thus assigns the element to one of the k clusters. Next, the centroids of each cluster are reevaluated and the elements are reassigned if necessary. The reevaluation/reassignment step is repeated until no element changes cluster and the clustering converges. (Alternatively, execution stops after a fixed number of iterations.)

The algorithm is guaranteed to converge: at each step the overall clustering is improved, therefore no configuration is repeated; since there are finite number of ways to allocate n elements among k groups, a local optimum is reached after a finite number of iterations. The complexity of K-means is $O(kn)$ with linear factors in the dimensionality of the vectors and the number of iterations (assuming a fixed number of iterations).

K-means is more efficient than hierarchical clustering in terms of computational resources, since k is usually a much smaller number than n. It is also conceptually simpler but it has limitations. One critical aspect is how we choose the initial centroids since K-means is sensitive to the starting configuration. If there is preliminary information about the cluster configuration, better than random initial selection can be made based on some kind of a heuristic method. For example, if the working data or data similar to it has already been clustered before, the algorithm can easily take advantage of the existing clusters by starting with their centroids. In case there is no a priori information, K-means could fail to find a good solution if it gets trapped in a local maximum depending on the initial condition.

K-means is also sensitive to *outliers*, extreme points which lie notably

far away from most of the points in the dataset. This is due to fact that K-means computes the centroid of points as their mean, and an outlier could pull the centroid away from its true position.

Finally, K-means requires that the final number of clusters is decided in advance and just like the initial centroids this is hard to determine without a priori knowledge. In this respect, agglomerative clustering algorithms have one important conceptual advantage over K-means. They take as an input argument a similarity threshold θ which determines at what level to cut the dendrogram to generate clusters. Clusters whose similarity is less than $1-\theta$ are not merged, so the process of merging can stop once this level is reached. An example is shown in Figure 2.5. This gives more flexibility as we only need to decide at what point the advantage of relative similarity is less than the disadvantage of relative dissimilarity, and hence there is no information gain in joining clusters any further.

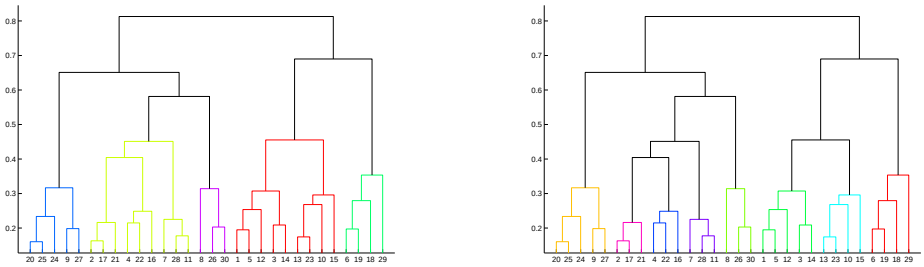
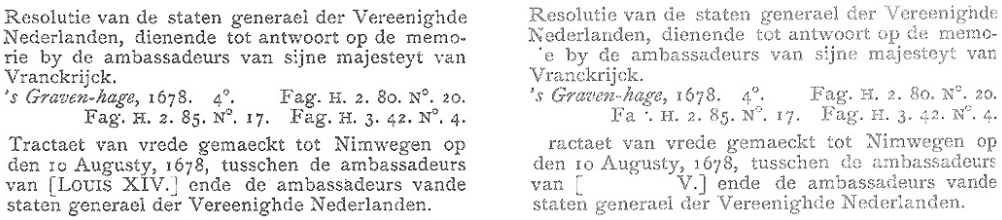


Figure 2.5: The dendrogram in Figure 2.2 cut at two different similarity thresholds. On the left, specifying the threshold at 0.5 results in creating five (and looser) clusters; on the right, specifying the threshold at 0.4 results in creating eight (and tighter) clusters.

Ultimately, the important distinction between the different kinds of clustering algorithms is the quality of the clusters they produce and this depends to great extent on the similarity structure of the space we want to partition. Therefore, there is not a universally applicable clustering technique which fits all data equally well.

2.9 Textual Images

Textual images are images of typed or typeset documents such as scanned or faxed pages. They are made up primarily of text although they may include some nontextual elements, e.g. logos, trademarks, signatures or drawings. Figure 2.6 provides an example of such an image. Although neither textual images nor compression is the subject of this paper, the efficient compressing technique described below resembles closely the image processing phase of automatic image annotation.


Resolutie van de staten generael der Vereenighde Nederlanden, dienende tot antwoort op de memorie by de ambassadeurs van sijne majesteyt van Vranckrijck.
's Graven-hage, 1678. 4°. Fag. H. 2. 80. N°. 20.
Fag. H. 2. 85. N°. 17. Fag. H. 3. 42. N°. 4.

Tractaet van vrede gemaeckt tot Nimwegen op den 10 Augusty, 1678, tusschen de ambassadeurs van [LOUIS XIV.] ende de ambassadeurs vande staten generael der Vereenighde Nederlanden.

Resolutie van de staten generael der Vereenighde Nederlanden, dienende tot antwoort op de memo- 'e by de ambassadeurs van sijne majesteyt van Vranckrijck.
's Graven-hage, 1678. 4°. Fag. H. 2. 80. N°. 20.
Fa '. H. 2. 85. N°. 17. Fag. H. 3. 42. N°. 4.

ractaet van vrede gemaeckt tot Nimwegen op den 10 Augusty, 1678, tusschen de ambassadeurs van [V.] ende de ambassadeurs vande staten generael der Vereenighde Nederlanden.

Figure 2.6: Textual image and its reconstruction after compression.

The compressing method utilizes the fact that textual images contain only or mostly text in the form of strings of characters [23]. By extracting connected groups of black pixels, a library of the distinct symbols occurring in the image is constructed. Figure 2.7 shows the symbol library generated from the textual image in Figure 2.6. The library is similar to an alphabet of shapes, which usually correspond to characters and digits. It is not necessary to recognize the symbols by determining which characters they stand for, as they can be processed and saved as bitmaps. This allows representing each symbol by a number that points to a position in the library. The compression is achieved by substituting repeated symbols with the same number and having one entry for all of them in the library. The number sequence together with the library can be used to restore the original image, though the reconstruction is an approximation rather than an exact copy of the original document since what we have described is a

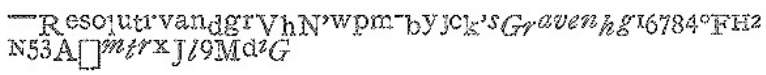


Figure 2.7: The symbol library built to represent the textual image shown in Figure 2.6.

lossy technique.

Of course, in general images are more complex than a digitized image of printed text. However, a similar technique can still be applied in order to represent images in terms of a library of symbols rather than visual features. This is discussed further in the next section.

2.9.1 Image Processing

The preliminary step of automatic annotation is to obtain an image vocabulary for the collection and use it to represent images in terms of the elements of a finite set of symbols describing visual components. First, an image is divided into regions and a set of visual features is extracted from each region. Those features may include RGB histograms (measuring amount of red, green and blue color) and LAB histograms (measuring color differences), color moments and texture gradients. The partitioning itself can be implemented either by applying a segmentation algorithm, e.g. Normalized cuts, or simply by dividing the image into a rectangular grid. Dividing into a grid of rectangles is fast and simple but a segmentation algorithm can be applied with the intention of getting semantically contiguous regions. However, Carbonetto *et al* report that the two approaches result in equal performance [5]. In fact, current segmentation algorithms often have poor accuracy and the process might split an object into several pieces or fail to separate two objects. Since segmentation has a higher computational cost and the naive approach does not penalize performance, it is more reasonable to use grid partitioning.

After partitioning, the regions from the entire collection are grouped together on the basis of the extracted visual features. The regions forming one group are represented by the same visual token.[4] This is analogous to the process of compressing textual images and the result is building a restricted vocabulary. But in contrast to the representation of a textual image, the token representation of an image is not structured: it is a bag of visual tokens in no particular order. Thus the synthetic visual language has no structures similar to the words and sentences making up the text captured in a textual image.

Once image processing phase is completed, the system relies on a set of training images that have been manually annotated to learn correspondences between words and visterms.

2.10 Automatic Annotation

Finally, we give a theoretical definition of the task we set to achieve - assigning suitable, descriptive words to images given their visual components. Those visual components, which we would call for now *visterms*, are abstract concepts for representing the visual appearance or features of images such as color, shape and texture. We explain how they themselves are generated in the next chapter. In the most general setting, visterms are similar to the symbols used to compress textual images.

Formally, we want to construct a model M that assigns words to each image I in an unannotated collection C. The model formulates its decision given the visterms of I and selects those words from a restricted vocabulary V that best describe the content of the image:

[4]Visual token refers to the identifier of a group of similar regions. There is not a standard term for this concept in the literature.

$$M : (I = \{v_1, ..., v_m\} \in C) \rightarrow (A = \{w_1, ..., w_k\} \in V)$$

To achieve this, we use a training collection T consisting of already annotated images J for which we know both the visterm representation $v_1, ..., v_m$ and the textual representation $w_1, ..., w_k$. In our dataset the number of visterms m is the same for all images in C and T, but in general images might have visterm representations of varying length (this is the case when segmentation is used to discretize images into regions).

The vocabulary V is restricted to include only words appearing in the annotations of training images because the model requires examples to learn how particular words and visterms relate to each other.

By assigning words to images, the annotation model in effect generates annotations. These annotations can be used by a text-based image retrieval system for finding relevant images given a set of query words. The system represents images indirectly by indexing their annotations instead of their visual features; given a specific query it performs straightforward text retrieval by comparing query and annotations to find matching words.

In the next chapter, we present three approaches that have been proposed for achieving this task. In particular, we introduce the Cross-media Relevance model, the generative language modeling technique that we extend by incorporating clustering.

Chapter 3

Previous Work

Learning how visual features and words are associated has many potential applications. For example, it can be applied in a straightforward way to create automatic image annotations, which in turn can be used to organize, browse and search image collections. A variety of approaches have been proposed for automatically assigning words to describe images [11].

3.1 Co-occurrence Model

In one of the earliest studies on automatic image annotation, Mori *et al* proposed using the co-occurrence of words and image regions to discover region-to-word relationships [18]. The procedure they developed includes the following steps:

1. Divide each training image into a rectangular grid and extract a set of visual features from each rectangular region.

2. Partition the space of feature vectors. Represent the regions corresponding to vectors belonging to the same subspace by the centroid of the subspace.

3. Assume each region inherits the words assigned to the whole image. Count the frequencies of words across regions and use word-region co-occurrences to estimate the probability of a word given a region.

4. To annotate a test image, divide it into a rectangular grid and extract the same visual features as for training images. For each region find the closest centroid from the training space and assign it to the group corresponding to that centroid. Take the word probabilities for all regions comprising the test image and average them to get the final probabilities. Use the n most likely words to construct an annotation of length n.

For the second step Mori *et al* use an incremental vector quantization algorithm that maps a set of vectors $f_i \in R^n$ into a finite set of centroids $c_j \in R^n$. The R^n space is partitioned among the centroids so that no two quantization regions overlap. The number of partitions is selected automatically in the mapping process but depends on an input parameter that specifies the threshold error: if the distance from a vector f_i to one of the centroids is smaller than the threshold value, then f_i is assigned to the respective partition and represented by its centroid. Otherwise, a new partition is created with centroid $c_j = f_i$.

To estimate the probability of a word w_i being associated with a centroid c_j, Mori *et al* use word-region co-occurrence statistics:

$$P(w_i|c_j) = \frac{m_{ji}}{\sum_{k=1}^{V} m_{jk}}$$

where m_{ji} is the frequency of w_i in the group represented by c_j, and V is the size of the vocabulary. The idea is to gather a lot of training statistics and use it to distinguish correct from incorrect region-to-word correlations. For example, consider an image of a tiger in the grass. Regions with grass

pattern inherit both 'tiger' and 'grass'. However, in other images 'grass' might appear in other settings, e.g. in a garden, so the particular region will inherit 'grass' and 'flowers'. By combining the information of just these two occurrences we have 'grass' twice, 'tiger' once and 'flowers' once. Thus 'grass' would get a higher probability than either 'tiger' or 'flowers'. Collecting more examples would help to get better estimates and give low probability to unsuitable words and high probability to appropriate ones.

The Co-occurrence model is simple and requires large amounts of training data to estimate the true probabilities. It is also biased towards assigning frequent words. (Rare words are probably more useful for retrieval as users would most often search for specific, interesting people, animals, places, etc.)

Several other models for automatic image annotation have been proposed, which all follow the general procedure outlined above: Start by dividing images into regions (either rectangular blocks or segments), then extract visual features and discretize the feature space by grouping similar regions into tokens. Next, compute the conditional probabilities $P(\textit{word} \mid \textit{visual token})$ for each pair of word and token. Finally, use those probabilities to rank words given an annotated image and select the highest ranked to describe it.

The definition of an automatic annotation model is independent of what visual features are chosen to process the images. Any combination of features can be used and the decision can be based on a heuristic or systematic approach, which tries to pick features maximizing overall performance. How the images are split into regions also affects performance without modifying the definition of the annotation model. Annotation models differ in the approach used to estimate the conditional probabilities of words given visual tokens and the specific details of how the probabilities for individual

image regions are combined to choose the most likely words for the image as a whole.

The next section describes a model which treats the assignment of word probabilities like translation from visual components to words.

3.2 Machine Translation Model (MT)

Duygulu *et al* model the process of assigning words to images as a form of multi-modal translation from visual tokens to words, which is similar to translation from one written natural language to another [8].

Statistical machine translation maximizes the conditional probabilities $P(e|f)$ where f is a word in the target language and e is a word in the source language. Training data consists of aligned bitext: portions of text such as paragraphs or sentences which are translations of each other. While it is known that the segments correspond in meaning, the exact correspondences between pairs of words are unknown. This is a missing data problem: given the correspondences we can estimate the translation probabilities and, vice versa, given the translation probabilities we can estimate the correspondences. The missing data problem can then be resolved with an unsupervised iterative technique called Expectation Maximization (EM). The EM algorithm iterates between the following two steps:

- Estimation: Compute the expected value of correspondences using the current value of the translation probabilities. (Start with the co-occurrence matrix on the first iteration.)
- Maximization: Revise translation probabilities using the new alignments.

The Machine Translation model considers the annotation and the blob representation of an image as aligned sentences and applies a classical IBM

statistical translation mechanism with blobs being the source language and words being the target language [4].[1] For each blob, the model learns which word it most probably translates into. One of the issues is that images and annotations are complimentary rather than interchangeable sources of information. With natural languages, when two sentences translate into each other, then they contain the same information. This is not exactly the case with words and visual tokens. Annotations might leave out obvious details, e.g. the color of a dog, and point to more interesting, semantic properties, e.g. its breed. Just like it is difficult to infer the color of a dog only from the word 'terrier', it is difficult to deduce its breed by considering color, shape and texture only. Also, most statistical machine translation techniques take into account the syntactic structure of a sentence to achieve better results. However, neither blob representation nor annotations have such structure.

The Machine Translation model assumes that there is a one-to-one correspondence between words and blobs, and therefore labels each region with the word that has the highest probability. The assumption, however, is not true in general. First of all, images have a different number of blobs and words in general. Also, words might not refer to a specific object and a region might be composed of (parts of) several objects.

In fact, the Machine Translation model attempts to associate each region with a particular word (this task is referred to as *region naming*), which is more ambitious than annotation. This task is close to object recognition, a very hard computer vision problem. However, region naming and object recognition are not necessary for creating annotations, though is the system is capable of correctly identifying the objects in an image, it would probably do a better job at annotating. The goal of automatic annotation

[1] In this paper, a *blob* denotes a region token derived after segmentation, and a *visterm* denotes a region token derived after grid partitioning.

is to associate words with the image as a whole rather than linking image regions with particular words.

The next section describes a technique which models the assignment of word probabilities as a discrete stochastic process.

3.3 Maximum Entropy Model

Maximum Entropy (MaxEnt) is a technique successfully applied to a variety of language tasks, including machine translation [3]. It is used to model a discrete stochastic process based on a series of observations. Knowledge about the process is incomplete as the algorithm relies on a set of training examples considered to be samples generated by the process. From the infinite number of models which satisfy the input constraints, the algorithm chooses the one with the highest *entropy* or uncertainty.[2] Thus the algorithm prefers a uniform distribution where no information is available, so as not to make additional assumptions.

Jeon *et al* apply a maximum entropy approach to automatic annotation [14]. They define two kinds of predicates which check for the co-occurrence of words and visterms and are weighted automatically by the MaxEnt algorithm. The weights are the conditional probabilities $P(word|image)$. Unigram predicates pair a word and a visterm. Bigram predicates pair two adjacent (vertically or horizontally) visterms and a word. The stochastic process takes an image as input and generates an annotation word as output.

Even complex low-level visual features have limited ability to identify objects. The Maximum Entropy model proposes a way to incorporate higher-level information by looking at image configuration. Therefore, this

[2]In information theory, entropy is more specifically defined as information content.

method has the advantage of taking into account, though in a simple way, how visterms are positioned relative to each other in contrast to the Machine Translation model, which considers the image as a bag of visual tokens. This is useful because certain concepts have the same placement with respect to one another in most images, e.g. sky always appears above grass.

The benefit of the Maximum Entropy model is incorporating additional, spatial information by using predicates of higher order. Although this might lead to performance improvements, it comes at the price of increased computational demands. Consider data consisting of m visterms and n words: there are mn unigram predicates, m^2n bigram predicates and m^3n trigram predicates. Obviously, there is a considerable increase in the number of model parameters, as the model tries to include more complex predicates. The other drawback is data scarcity. Even for a large collection, many combinations of visterms and words would appear hardly ever and there would be not much information for the algorithm to make good parameter estimations. This problem would quickly aggravate for predicates of higher order. Therefore, the Maximum Entropy model does not scale well.

In the next section we introduce a language modeling technique, which considers automatic image annotation as a retrieval problem and ranks words in terms of their relevance to the unannotated image.

3.4 Cross-Media Relevance Model (CMRM)

Jeon *et al* adapt a cross-lingual retrieval method for automatic image annotation [13]. To predict the underlying concepts of an unannotated image, their Cross-media Relevance model computes the joint probability of ob-

serving a word w and the blobs of the image together, for all words in the vocabulary. The joint distribution is estimated by comparing the test image with training images that contain w and a fixed number of the words with the highest probability are included in the annotation. Rather than assuming that particular words correspond to particular blobs as the Machine Translation model does, the Cross-media Relevance model uses blobs to measure the similarity between two images, assuming instead that images which are similar in terms of blobs are also similar in terms of their annotations. Therefore, this technique does not associate words and blobs directly but provides context for individual blobs - thus helping to disambiguate blobs based on the other blobs forming the image. Another advantage is that it is capable of performing ranked retrieval in addition to image annotation. Lavrenko *et al* have also proposed a modification of the Cross-media Relevance model that directly uses continuous-valued feature vectors rather than discretizing them into blobs [15].

The Cross-media Relevance model is a generalization of a text retrieval technique, which uses the query to estimate word probabilities in the class of relevant documents. As its name indicates, CMRM retrieves across media: the query is in one medium and the documents are in another. In the case of images, for example, the query is a word and the document is an image and the degree of similarity measures the likelihood of annotating the image with the word.

3.5 Cluster-Based Text Retrieval

Document clustering within the framework of language modeling for text retrieval has been investigated by Liu *et al* in [16]. They define two cluster-based methods: Cluster Query Likelihood (CQL), which builds language

models for clusters instead of documents, and Cluster-based Document model (CBDM), which smooths the language models of documents with the model of their respective cluster.

The motivation behind the cluster-based approach to information retrieval in general is exploiting corpus structure: how are documents related to each other given term co-occurrence patterns. Clustering utilizes collection-wide features that are ignored by individual-document analysis (such as TF×IDF weighting). Liu *et al* propose cluster-based language models for full-text retrieval which explore across-document word co-occurrence patterns in addition to within-document occurrence patterns.

CQL ranks clusters based on the probability of generating the query. First, document-based retrieval is performed and then the 1000 highest-ranked documents are clustered using an agglomerative clustering method. The resulting clusters are ranked according to $P(Query|Cluster)$ and the documents within a cluster are ranked according to $P(Query|Document)$, which is already computed in the first step. In CQL clusters play primarily a ranking role as they are directly used to estimate the conditional probabilities, which determine the level of relevance. Liu *et al* experiment with different hierarchical clustering algorithms, different threshold values for the clustering and different smoothing methods on various collections, and their results show that CQL is as effective as document-based retrieval.

CBDM ranks documents as in a straightforward document-based retrieval but smooths the probability estimates of words given documents with the cluster frequencies (and the cluster frequencies themselves are smoothed with the background collection frequencies). In CBDM clusters play primarily a smoothing role as they are indirectly used to smooth the language models of individual documents. Again, Liu *et al* experiment with different clustering techniques (both static and query-specific),

different clustering parameters and different smoothing methods on various collections, and their results show that LM retrieval, i.e. estimating $P(Query|Document)$, with CBDM performs significantly better than with the standard document model, and RM retrieval, i.e. estimating $P(Query, Document)$, with CBDM performs as well as with document model.

The experiments in [16] are extensive: they use six different document collections (one for parameter setting and five for testing) and therefore their results convincingly demonstrate that cluster-based language models used either for ranking or smoothing are at least as good and sometimes significantly better than document models. This implies that the similarity structure of the collection is a potentially useful source of information because clusters provide more representative statistics for term distribution as a result of combining multiple similar documents. Clusters, considered as longer "documents", include much more observations and this allows for a better approximation of their language model.

The investigation of Liu *et al* also brings up the question whether clustering would be effective in improving the performance of language models for image annotation. In the next chapter we discuss how CQL and CBDM are applied to the image annotation task, and explain how smoothing is used to refine estimations which are based on a limited set of sample observations.

Chapter 4

Cluster-Based Annotation

This work explores the hypothesis that clustering information can effectively improve automatic image annotation and hence image retrieval. We extend the Cross-media Relevance model to analyze the structure of an image collection and exploit statistics about groups of similar images in addition to statistics about individual images when estimating relationships between words and visterms. We believe that incorporating cluster-based information will improve the effectiveness of a system which takes into account only information derived from single images. To test our hypothesis, we propose two cluster-based models that define the Cluster Query Likelihood and Cluster-based Document model in an image retrieval context, and compare their performance with that of an unclustered image-based annotation model. We describe the models in this chapter and then we discuss implementation and experimental setup in the next chapter. For reference, we keep the names CQL and CBDM (although Cluster-based Image model is perhaps more appropriate in our case). We also keep in mind that text collections and image collections have different characteristics (including average size of the collection and the vocabulary and average length of individual documents). Therefore, when the documents are im-

ages instead of text, model parameters would most probably have different optimal values and the models in general might exhibit different behavior.

The two cluster-based models we propose, CQL and CBDM, as well as the original model compute the joint distribution of words and visterms. Given a particular image, these probabilities are used to rank potential annotation words in terms of their likelihood of being sampled from the conditional probability distribution given the visterms of the image. This joint distribution of words and visterms is the *relevance model* referred to in the name CMRM. A fixed number of the highest-ranked words are finally selected as the actual annotation.

4.1 Cross-Media Relevance Model (CMRM)

As explained in Section 3.4, Relevance modeling (RM) assumes that the words and visterms representing an image are generated stochastically from the same underlying probability distribution. For text retrieval, think about the query and a relevant document as two articles of different length written on the same topic - one is a very brief summary and the other is a full-text elaboration. The goal of RM is to model the underlying topic and use its model to estimate how likely it is that both the query and the document are generated by this same model.

When the documents are images, RM assumes that we can sample words and visterms from a joint probability model. This assumption seems justified because words and visterms are naturally related - after all, they describe the same concepts in two different ways. Think about a person who is reading a book about the jungle and comes across a paragraph describing a tiger in the grass. If she has some spare time, she can write down a few words summarizing the passage; she can also draw a picture of

the animal and its surroundings as she imagines it. The summary and the picture both represent the same idea as it is expressed in the book passage.

Therefore, we can think of the relevance model of an image I as a black box containing all the visterms that could possibly appear in I as well as all the words that could appear in its annotation (the box can contain the same word or visterm multiple times). We do not know what exactly the black box contains but the actual visterms representing the image are observations obtained by sampling m times from the distributions $P(\cdot|I)$, i.e. by randomly taking m objects out of the black box.[1] To annotate the image, we need to select words but we cannot sample directly from the black box because the distribution $P(\cdot|I)$ is unknown. Our best strategy is to approximate it using the visterms. Returning to the jungle book analogy, we do not know the passage that the person read but we have the picture she drew and we want to guess what summary she wrote.

$$P(w|I) \approx P(w|v_1, ..., v_m)$$

The visterm representation of an image contains no words, therefore the maximum likelihood estimation for each word in the vocabulary would be equal to zero. Maximum likelihood is defined as $P_{MLE}(a_i|A) = \frac{\#(a_i,A)}{|A|}$, where $\#(a_i, A)$ denotes the number of occurrences of a_i in A and $|A|$ denotes the overall number of elements in A. Because we cannot directly use the visterms of I to estimate maximum likelihoods for words (they are all zero), we use the training collection T to estimate the joint distribution of observing a candidate word w and the visterms of I together, assuming identical and independent distribution of words and visterms (i.i.d.).

First, we compute the conditional probabilities $P(w|J)$ and $P(v|J)$ for

[1]The notation $\cdot$ stands for either a word w or a visterm v. Remember that we assume that the model of an image contains both words and visterms.

each training image J:

$$P(w|J) = (1-\alpha_J)P_{MLE}(w|J) + \alpha_J P_{MLE}(w|T) \quad (4.1)$$

$$P(v|J) = (1-\beta_J)P_{MLE}(v|J) + \beta_J P_{MLE}(v|T) \quad (4.2)$$

We approximate $P(\cdot|J)$, where $\cdot$ stands for a particular word or visterm, with a smoothed maximum likelihood. To estimate the maximum likelihood $P_{MLE}(\cdot|J)$, we count how many times the term appears in the representation of a training image and normalize that count by dividing by the total size of the representation. However, for terms that do not actually appear in image J we have $P_{MLE}(\cdot|J) = 0$, which effectively renders associating the term with J entirely impossible. This should be avoided as it means that the estimated probability distribution is unreliable, especially for the images we work with because they have very short representations.

To ensure that terms which do not occur in the representation of an image have nonzero probability, we take some probability mass from words that do occur and distribute it among those that do not. We achieve this by smoothing the maximum likelihood estimates with the general relative frequency as computed from the entire collection T. Smoothing is discussed in Section 4.1.1.

We can think of the work so far as a preliminary phase. The set of training images is the database containing all available information about word-visterm correlations, but we have calculated only probabilities for individual words and visterms. In isolation these do not say anything about what interrelations exist between words and visterms, so we cannot annotate images just yet. It is a specific property of the Cross-media Relevance model that the computation which actually finds such correlations is per-

formed at run-time only after the image to be annotated is provided. The exact formula is

$$P(w, v_1, ..., v_m) = \sum_{J \in T} P(J)P(w|J) \prod_{i=1}^{m} P(v_i|J) \tag{4.3}$$

where the prior probabilities $P(J)$ are kept uniform because the training images are equally likely. Thus, we compute word probabilities after we are given the visterms of the unannotated test image.

The method essentially compares the visterms of a test image I with the visterms of J for all Js in the set of training examples, implicitly ranks the Js in terms of their visual similarity to I and finds what words co-occur with the visterms of similar Js. In IR terms, the Cross-media Relevance model ranks training images in terms of their relevance to the image we want to annotate, and looks at the annotations of relevant images because these should be similar to the annotation we want to estimate. The approach is based on the assumption that we can use available knowledge, in this case information extracted from already annotated images, to annotate new ones. Rather than recognizing explicitly the objects in an image and assigning the corresponding words, we build a statistical model which estimates the likelihood of words being used to describe an image.

To understand how CMRM ranks images, consider again Equation 4.1. The part where we compare I and J in terms of their visual similarity is $\prod_{i=1}^{m} P(v_i|J)$. First, we look at the individual probabilities $P(v_i|J)$ for the m visterms of I being sampled from the model of J: if I and J are similar, those probabilities are high. Their product quantifies the degree of similarity as a single number, which amounts to ranking the Js: for similar Js this value is higher than for dissimilar Js. The probability $P(w|J)$ determines the how likely it is to associate w with J. Finally, by multiplying

$\prod_{i=1}^{m} P(v_i|J)$ and $P(w|J)$ we decide how much weight to assign to the word w when considering what words to select for the annotation of I.

Following this approach, we estimate a relevance model $P(w, I)$ of observing a word and a set of visterms together. In contrast, the Machine Translation model (MT) computes word probabilities given a particular visterm, which can be conveniently precomputed and stored before we start to annotate. (MT is discussed in Section 3.2.) With the Cross-media Relevance model, it is impossible to precompute $P(w, v_1, ..., v_{24})$ because there are enormous number of ways in which 500 visterms can be combined to form an image of 24 visterms - 500^{24} to be exact. However, this does not mean that a retrieval system based on CMRM is slower than one based on MT. In the overall design of the retrieval system, the annotation phase would be completed before the system is finally in use - all unannotated images that users are to access will be annotated one by one before making the system available. Annotations are only proxies that enable the system to search for images based on text and annotation itself is an intermediary process.

Since there is no prior knowledge about the objects in an image, the system does not have any indication about what words might be appropriate for it. Without making any assumptions, it can compute $P(w, I)$ for each word in the working vocabulary. Those probabilities reflect the likelihood of assigning a particular word w to the image with respect to its visual components. Finally, the system ranks words in terms of these likelihoods and creates an annotation of the desired length by selecting the highest ranked words.

4.1.1 Smoothing

We already briefly mentioned smoothing in the previous section. We discuss it here in greater detail because smoothing is crucial in relevance modeling and in language modeling in general.

Smoothing is the process of adjusting the maximum likelihood estimates of events to guarantee that no event is made entirely impossible by giving it zero probability. The computation looks intuitive but it in fact overestimates seen events and underestimates unseen ones. Because a maximum likelihood estimate is computed in terms of the actual number of occurrences, MLE of an unseen event is of course zero.

The role of smoothing is to correct the maximum likelihood estimates by taking some probability mass from events that do occur and distributing it among those that do not. In the case of images, a "seen event" is to observe a word or a visterm in the representation of an image. To understand why MLEs over-represent seen events at the expense of unseen ones, consider the following situation: Because annotations are only a few words long, we may decide to extend them by adding more words. If we use MLEs, it would turn out that all words which are not already included have the same probability - zero. We would not be able to make an informed decision about which words to select next and our best bet would be to choose randomly.

The accuracy of maximum likelihoods estimation, i.e. how close MLEs are to the true probabilities, depends to a great extent on the size of the observed data. Since MLEs are only approximations of the true values, the more observations, the better approximation. In our case, the observations for an image are its words and visterms: 24 visterms and from 1 to 5 words for a combined sample of 25 to 29 observations. Thus, the sample from the underlying probability distribution, which we are trying to approximate,

is very limited. Consequently, for any image, most words and visterms do not appear at all in its representation. However, the fact that a term has not occurred in the first 25 observations does not imply that it does not have even the slightest chance to occur if we continue to draw terms out of the "black box". For this reason, it is essential not to rule out terms with absolute confidence - it is wiser to say that a term is highly unlikely (very small probability) than to say that it is entirely impossible (zero probability). To this purpose, some probability weight is taken away from the higher P_{MLE} words and allocated among those with zero P_{MLE}. Different smoothing methods do this in a different way, usually using the background frequency for better approximation. For images, smoothing has a significant effect because the image representations are so short.

All smoothing techniques define two distributions - one for seen events, and the other for unseen events. For a general situation, where we want to build models for the documents D in a collection C, smoothing gives the following:[2]

$$P(t|D) = \begin{cases} P_s(t|D) & \text{if the term } t \text{ occurs in document } D \\ P_u(t|D) & \text{otherwise} \end{cases}$$

Having no knowledge about the distribution of unseen terms, we can use the collection to compute general frequencies and assume that the frequency of an unseen term is proportional to its general frequency.

$$P(t|D) = \begin{cases} P_s(t|D) & \text{if } t \text{ occurs in } D \\ \alpha_D P(t|C) & \text{otherwise} \end{cases} \tag{4.4}$$

[2]The probabilities of seen and unseen events together should sum up to 1. Since we decrease the probability of seen events a little bit, so that we can distribute the difference to unseen events, we know that $P_s(t|D) < P_{MLE}(t|D)$.

where the coefficient α_D is a general symbol of smoothing which has different form in different techniques and controls how much probability mass is assigned to unseen events and how it is allocated, so that the probabilities sum up to 1. As the subscript indicates, α can be document-dependent.

Studies have shown that the performance of retrieval methods based on language modeling is highly sensitive to the choice of smoothing technique and the setting of smoothing parameters [24]. In our experiments we alternatively use the two most popular smoothing methods in language processing and information retrieval - Jelinek-Mercer smoothing and Bayesian smoothing with Dirichlet prior. The difference is in how the probability mass taken from seen events is allocated among unseen events.

Jelinek-Mercer Smoothing

This smoothing technique involves linear interpolation between the document model $P_{MLE}(t|D)$ and the collection model $P_{MLE}(t|C)$. It is also referred to as *linear* smoothing.

$$P(t|D) = (1 - \lambda)\frac{\#(t, D)}{|D|} + \lambda\frac{\#(t, C)}{|C|}$$

Here the smoothing parameter α_D from Equation 4.1.1 is simply λ. It controls the influence of background frequencies with values ranging from 0 to 1, and has the same value for each document. The bigger λ, the more smoothing.

Bayesian Smoothing with Dirichlet Prior

With this smoothing technique the estimation is obtained via Bayes rule given the prior probability of the distribution.

$$P(t|D) = \frac{\#(t, D) + \lambda \frac{\#(t,C)}{|C|}}{|D| + \lambda}$$

In this case α_D has a more complicated form: $\alpha_D = \frac{\lambda}{|D|+\lambda}$. This value varies with the length of documents because their size $|D|$ determines how big the denominator is. Thus longer documents are penalized more, which effectively implements length normalization.

The parameter λ is called the *hyper-parameter* and it acts as a pseudo (virtual) count added to the actual number of occurrences for each term. Values of λ are integers ranging from 1000 to 5000 for full-text retrieval. Again, the bigger λ, the more smoothing.

Since Bayesian smoothing implicitly involves length-normalization, it usually performs better than Jelinek-Mercer in IR tasks, where document collections are typically heterogeneous and contain documents of various lengths [24]. In general, the decision what smoothing technique to use should depend on the task and the characteristics of the data we are modeling.[3]

In Section 2.1 we introduced TF×IDF weighting and explained how term frequency and inverse document frequency capture two different and complimentary sources of information about the relative importance of a term. At first glance, the language modeling approach seems to be fundamentally different from the TD×IDF metric commonly used by most IR

[3]Actually, the definition of Jelinek-Mercer is more general. It involves using $(n-1)$-grams to smooth n-grams; this implies we need $n-1$ lower-order linear interpolation estimations to smooth an n-gram [12]. In fact, it is one of the most popular smoothing techniques in LM. Here, we only provide the simplest definition for $n = 1$ because we build unigrams and we do not need the more complex general form. However, keep in the mind that we cannot claim either Bayesian or Jelinek-Mercer smoothing to be universally superior to the other. They are conceptually different and one would be more appropriate than the other in specific situations.

systems, regardless of the underlying retrieval model. Specifically, it seems that LM looks only at TF and ignores IDF, which is alarming because IDF reflects the fact that common words are less informative and have less power to discriminate between relevant and nonrelevant documents. However, a more sophisticated analysis of language modeling shows that smoothing plays a role very similar to IDF weighting by using both language models of individual documents and a language model of the whole collection to estimate word probability distributions. In [24] Zhai *et al* demonstrate that when using collection frequencies for smoothing, the probability of a matched query term is directly proportional to the document term frequency and inversely proportional to the collection frequency. This result not only proves that LM implicitly exploits the across-collection occurrence patterns of a term, but also provides a justification for the heuristic TF×IDF technique.

4.2 Cluster Query-Likelihood Model (CQL)

To understand better the weaknesses of the Cross-media Relevance model as well as the reasons why clustering could improve it, we first need to consider the limitations of the image representation we work with. How to deal with these limitations and what assumptions are made to achieve this is what actually distinguishes the different methods discussed in Chapter 3.

For example, the manual annotations necessary to train the automatic system are supplied by humans and therefore are intrinsically subjective. Even with strict guidelines, different people might choose different terms to describe objects that appear very similar. One issue arising from this fact is that huge sets of manually annotated images would be inconsistent because

assigning annotations would be a collaborative work assembled over time. People also have different understandings of what is an 'important' object and therefore should be directly referenced in the annotation, and what is a 'trivial' or 'insignificant' detail and therefore could be left out. So another result of subjectivity is that some annotations are incomplete - the annotation does not indicate one of the objects in the image. Sky, for example, is so common that it is present in almost every picture in our collection.[4] Of course, this does not remove the sky from the image itself, so it is still accounted for in the visterm representation. In such cases, CMRM would fail to detect valid co-occurrences of light-blue patch with the word 'sky'.

The visterms, on the other hand, are a somewhat artificial device to capture the visual content of an image. (In fact, many questions can be raised about the theoretical foundations and applicability of this approach. However, we do not examine any of those here, because all annotations models we have discussed use the same representation and therefore face the same problems arising from representation constraints.) After all, images are extremely rich in detail and they are not structured in the rigid way that a rectangular grid presumes.[5] When a person looks at a picture or a photograph, she certainly does not see a group of rectangular patches of color, shape and texture. After images are divided into regions, additional detail is lost when extracting the visual features, and then still more information is lost when visual features of various regions are clustered into visterms. And because these procedures are performed automatically, some error is introduced at each level of analysis.

---

[4]The images are almost exclusively wildlife or outdoors photographs. More detail is provided in Section 5.2.

[5]For this reason, segmentation should produce a more natural and intuitive representation. However, existing segmentation algorithms fail to produce segments with the necessary quality.

The examples above are instances of a bigger, more profound problem which CMRM cannot handle - that there is a distinction between the appearance and the semantics of an image. Images could be not at all alike on a visual level and at the same time very similar on a higher, conceptual level. Since CMRM estimates similarity between images based on their visterms, this poses a significant problem.

To deal with this issue and improve the Cross-media Relevance model, we take advantage of cluster statistics, thus compensating for the limitations of the image representation we work with. We obtain additional information from similar images that allows us to get better approximation of the word and visterm distributions, and therefore we estimate the relevance models better. In short, we try to find word-visterm co-occurrences not only directly from individual images but also indirectly from similar images, e.g. visterms which co-occur with visterms which co-occur with a word become somewhat associated with this word even if it is missing from the original annotation. And if it is present, the association is reinforced. Thus we can compensate for missing information and reduce differences between related images.

Following the ideas of the Cross-media Relevance model, we compute the joint distribution $P(w, I)$ treating the clusters G as if they were large images and keeping the prior probabilities $P(G)$ uniform:[6]

$$P(w, v_1, ..., v_m) = \sum_{G \in T} P(G) P(w|G) \prod_{i=1}^{m} P(v_i|G) \tag{4.5}$$

To find the conditional probabilities $P(w|G)$ we count how many times the word w appears in the cluster G. The maximum-likelihood estimates

[6]For consistency we use the following notation: C for the set of test images, T for the set of training images, G for a cluster (or group), I for a test image, and J for a training image.

are smoothed with the collection frequency. We compute $P(w|G)$ in a similar way.

$$P(w|G) = (1-\alpha_G)\frac{\#(w,G)}{|G|} + \alpha_G\frac{\#(w,T)}{|T|} \quad (4.6)$$

$$P(v|G) = (1-\beta_G)\frac{\#(v,G)}{|G|} + \beta_G\frac{\#(v,T)}{|T|} \quad (4.7)$$

4.3 Cluster-Based Document Model (CBDM)

A logical extension of the idea that clusters are a source of useful information is to use clusters for generalization of co-occurrence patterns. Recall that CMRM has to smooth word and visterm frequencies, so that the probabilities of unseen words and visterms are not underestimated, and that smoothing techniques use a background (also called *fallback* or *back-off*) distribution to adjust the models.

A very trivial fallback method is to assume a uniform distribution but this is not realistic, since in any collection, some terms tend to occur more often than others. The non-uniform distribution of terms within the collection itself is another source of generalization, which is immediately available and is moreover derived from information actually contained in the images.

Clusters are an alternative source of generalization. Less general than the entire collection and more specific to a subset of images, clusters are more sensitive to the occurrence patterns of individual words, especially rare, non-uniformly distributed words. For example, a word might occur only a few times in the whole collection with most of those occurrences concentrated in a certain group of images. In this case, clusters might be more appropriate for smoothing image models with respect to that par-

ticular word since the cluster combining the relevant images gives a high probability of the word while all other clusters keep the probability low. For example, word and visterms distributions in wildlife images would be different from word visterm distributions in city images, so they should be computed separately even is the same collection contains both kinds of images.

In this cluster-based annotation technique we use the training images J instead of the clusters G in estimating the joint probability of observing words and visterms together:

$$P(w, v_1, ..., v_m) = \sum_{J \in T} P(J) P(w|J) \prod_{i=1}^{m} P(v_i|J)$$

The above equation is exactly the same as Equation 4.1. The difference is in estimating the conditional probabilities $P(w|J)$ and $P(v|J)$. In CBDM we smooth the models of individual documents with the model of their respective cluster while clusters themselves are smoothed with the collection:

$$\begin{aligned} P(w|J) = (1-\alpha_J)\frac{\#(w,J)}{|J|} + \\ \alpha_J \left((1-\alpha_{G_J})\frac{\#(w,G_J)}{|G_J|} + \alpha_{G_J}\frac{\#(w,T)}{|T|} \right) \end{aligned} \tag{4.8}$$

$$\begin{aligned} P(v|J) = (1-\beta_J)\frac{\#(v,J)}{|J|} + \\ \beta_J \left((1-\beta_{G_J})\frac{\#(v,G_J)}{|G_J|} + \beta_{G_J}\frac{\#(v,T)}{|T|} \right) \end{aligned} \tag{4.9}$$

In the previous two sections, we defined the two cluster-based models to be implemented and evaluated. With our investigation we bring up two main questions: Are clusters better than images for learning word-visterm

co-occurrences? Are clusters better than the collection for making generalizations? We ask those questions to corroborate our hypothesis that clusters provide meaningful information, which is otherwise not present in the analysis of individual images. We will show that clustering does increase recall and precision and this is extremely important - it implies that clustering helps us to learn from the collection structure and obtain additional knowledge even when we do not have any feedback or prior knowledge about the collection we work with, which is most often the case. And from a broader perspective, our models do not depend on the particular image representation in terms of words and visterms, and therefore are general enough to be applied for analyzing any kind of documents represented in two different, yet complementary, formats.

Chapter 5

Experimental Setup

In the previous chapter we presented CQL and CBDM, the two cluster-based models we propose and investigate; in the current chapter we discuss our methodology and experimental setup.

5.1 Clustering Images

In order to extract information from groups of similar images, we need to first partition the set of training images into clusters. Since the Relevance model assumes that words and visterms are generated by the same joint unigram distribution, it makes sense for the clustering algorithm to take into account both words and visterms rather than only words or only visterms, when computing distances between images. As discussed in [2], the main advantage of clustering on both words and visterms is that people perceive both the visual and semantic content of images. For example, the user might be interested in pictures of red flowers. However, since colors are inherently present in images, annotators would probably not assign attributes such as 'red', 'green', 'blue', etc., explicitly.[1] Therefore, by combining visual features and text, the clustering algorithm could generate

[1] Of course, color is meaningless for black-and-white images.

clusters with both visual and semantic coherence.

Before clustering we use Latent Semantic Indexing (LSI), described in Section 2.5, to map the vector representations of images into a lower-dimensional concept space. First, we generate a $(N_v + N_w)$-dimensional count-vector c_i for each training image i, where N_w is the size of the word vocabulary and N_v is the size of the visterm vocabulary. In our collection there are 371 distinct words and 500 distinct visterms, therefore $N_w = 371$ and $N_w = 500$. The count-vectors reflect the number of occurrences of words and visterms in the textual and visual representations of an image: $c_i[j]$ = frequency of j in the representation of image i. For $j = 1, ..., 371$ the element j is a word; for $j = 372, ..., 871$ it is a visterm.

At this point, the coordinates explicitly reflect term frequency.[2] To take into account global term usage as well, we apply TF×IDF weighting on the counts. Recall that IDF stands for Inverse Document Frequency and it reflects how often a term appears across the entire collection. IDF penalizes more frequent terms as they are less helpful in differentiating between relevant and nonrelevant documents.

Next, we apply LSI in order to extract information about global co-occurrence patterns, which reflect how terms are related to each other across the collection. LSI considers documents which have many terms in common to be semantically similar and brings them closer together in the LSI space; it considers images which have few terms in common to be semantically different and brings them further apart in the LSI space. We already emphasized that clustering performance depends on the configuration of the document space. Therefore, our hypothesis is that applying LSI on the image vectors prior to clustering improves the quality of the constructed clusters.

---

[2]Recall that we refer to words and visterms jointly as terms.

After performing LSI and clustering, we combine the images in the same cluster by grouping their words on one hand and their visterms on the other hand, and without removing duplicates. The LSI representation where documents are represented as vectors is used only for the clustering phase. Once the clusters are created by assigning a particular cluster label to each training image, the reduced co-occurrence matrix is no longer used. In fact, index matrices are not used in language modeling in the first place because this family of methods encompasses a variety of statistical probabilistic techniques that estimate probability distributions rather than weighted occurrences using TF×IDF or another weighting scheme.

5.2 Dataset

For our experiments we use a portion of the Corel Stock Photo library [20]. This is a collection of high-resolution color photographs grouped according to specific themes into CDs of 100 images each. We use 50 CDs; of the 5000 images, 4500 are for training and the rest are for testing. Each image is assigned a manual annotation consisting of 1 to 5 words. For the 5000 images, there are 371 distinct words. This is the same data used in [8] and [13] but the images are discretized using a rectangular grid instead of a segmentation algorithm, and hence into visterms instead of blobs.

The actual image processing has been performed at the Multimedia Retrieval Lab, Center of Intelligent Information Retrieval, University of Massachusetts, Amherst. The images have been divided into a grid of 4-by-6 rectangular blocks. Vertical images have been divided into a grid of 6-by-4 rectangular blocks. Color features (standard deviation and skewness of the RGB values, and standard deviation, skewness and average of the CIE-Lab values) have been extracted from each of the 24 blocks of an

image. K-means clustering with $K = 500$ has been used to partition the visual feature space and cluster regions into visterms. Details can be found in [1] and [13].

The purpose of clustering feature space is to represent images using a finite discrete set of visual components. We want to emphasize that clustering for obtaining discrete image representations has nothing to do with clustering for finding groups of similar images. This is an example of applying the same general technique for two different tasks: one is to generalize regions in image space with similar visual features, the other is to generalize images with similar visterms and annotations. In theory, we are not interested in how image representations are created. Had a different approach been used to generate the visterms, Cluster-based Relevance models would still be used in exactly the same way. The definition of an automatic annotation model is independent of the particular image representation, although performance depends on its quality.

5.3 Parameter Setting

The cluster-based models apply a lot of smoothing in their estimations - the parameters $\alpha_J, \beta_J, \alpha_G$ and β_G in Equations 4.6, 4.7, 4.8 and 4.8 are all general symbols for smoothing. To avoid confusion when discussing the parameters and their optimal values, we give each parameter its own unique name and consistently use those names throughout the rest of this text. We refer to the smoothing parameter for words in image models as α, the smoothing parameter for visterms in image models as β, the smoothing parameter for words in cluster models as γ, and the smoothing parameter for visterms in cluster models as δ.

The two cluster-based methods CQL and CBDM involve an additional

clustering parameter θ, which determines the size and therefore implicitly affects the quality of the clusters. Of course, when we partition a given set of elements, the more clusters we create, the fewer elements fall into the same cluster. Therefore, the fewer clusters, the greater diversity among cluster members. In K-means θ directly specifies the number of clusters. In agglomerative clustering θ specifies the similarity threshold at which we stop merging clusters. The threshold in turn determines at what level the dendrogram is cut to produce clusters: the lower in the tree, the more and smaller clusters; the higher in the tree, the fewer and bigger clusters.

The smoothing parameters also affect aspects of the model and therefore influence its performance. For example, α determines how much we rely on word frequency in an individual annotation to approximate the underlying model of an image. The bigger α is, the more we discount what words actually appear in the annotation and fall back to the general collection frequencies. Thus, as α increases image models and the corresponding probability distributions move closer to the background distribution and therefore they become more and more similar to each other. As a result of smoothing out the individual frequencies, the model becomes strongly biased in favor of the words which appear most often in the collection. In annotation this would have the effect of assigning the same words over and over again - the most frequent ones - to every test image.

Since we do not have additional data on which to estimate the smoothing parameters, we have to use the 4500 training images for both training and parameter setting. We apply 10-fold cross validation by dividing the 4500 images into 10 subsets of equal size and optimizing on some evaluation measure. For a given set of smoothing parameter values, we train the model 10 times using one of the folds for testing and the rest for training,

and we average the results of the 10 different trials. We choose the parameter setting that maximizes the average result. This strategy requires 10 times as much computational time but it is more reliable than the simpler holdout method which divides the training set into two subsets and trains the model only once. Evaluation based on the holdout method can strongly depend on the particular division, i.e. which images are used for training and which are used for validation. Evaluation with K-fold cross validation has less variance because it uses each image for validation exactly once (and for training $K-1$ times).

All three methods require tuning at least two parameters: CMRM - α and β; CQL - γ, δ and θ; CBDM - $\alpha, \beta, \gamma, \delta$ and θ. During optimization we do not need to train all combinations of parameter values exhaustively in order to find the optimum (and in practice this is virtually impossible since there are too many combinations of different values to try out). Because the parameters are independent of each other, we can set all of them but one to a certain value and then vary that one to find a local maximum. For example, with CQL we start by setting $\beta = \beta_{start}$ and $\theta = \theta_{start}$ and vary α. We run cross-validation on the 10 folds into which we have divided the training set optimizing on the chosen evaluation measure which specifies how we decide which value of α gives the highest performance. (The evaluation measure in question is the F-measure, described in the next section.) The starting values $\beta_{start}, \gamma_{start}, \delta_{start}$ are chosen with respect to the particular smoothing technique. For example, with Jelinek-Mercer smoothing we start at 0.5 which gives equal weight to the document frequencies and the collection frequencies.

Next, we fix $\alpha = \alpha_{max}$, keep $\theta = \theta_{start}$ and vary β. Again, we cross-validate on the 10 folds to select the best value of β. Finally, we keep $\alpha = \alpha_{max}$, set $\beta = \beta_{max}$ and vary θ. Another cross-validation gives us the

optimal clustering parameter. Then we put the 10 folds back together to obtain the full set of training images and train the model one last time with $\alpha_{max}, \beta_{max}$ and θ_{max}. Finally, we are ready to evaluate performance.

We follow the same procedure to train CBDM, the only difference is that it has more parameters to tune. With CQL we optimize γ, δ, θ in this order; with CBDM we optimize $\alpha, \beta, \gamma, \delta, \theta$ in this order.

5.4 Evaluation

Recall from Section 2.3 that in order to estimate recall and precision for a given query we need to know what the relevant documents actually are, i.e. we need relevance judgments. Similarly, to evaluate automatically generated annotations we compare them with the "true" annotations. To achieve this, we consider manually created annotations to be the ground truth. For evaluation we can gather a set of images and have someone sit down and annotate them. Or alternatively (and this is what we did), we can set aside a portion of the training images and use them for testing, since we already have manual annotations for these. (Of course, we should not train the models on the test images because results would be biased and deceptively better than what we would get in general.) And we will define an image to be relevant to a word if its manual annotation contains that word.

As discussed in the previous section, before being able to evaluate the performance of a model, we first have to train it by finding the optimal values of its parameters. And to tune parameters, we need to choose an evaluation measure for optimization.

5.4.1 F-Measure

The probabilistic models we work with involve smoothing maximum likelihood probabilities and therefore include smoothing parameters. As discussed in Section 2.3, there is an implicit tradeoff between recall and precision. Therefore, we should not use either measure for setting model parameters: we do not want to optimize on recall at the expense of precision or vice versa. Therefore, we optimize on the F-measure, a single comprehensive measure which combines the two. It is defined as the harmonic mean of precision and recall:

$$F = \frac{2 \cdot recall \cdot precision}{recall + precision}$$

We use the F-measure only during training. As a single quantity, it cannot illustrate how recall and precision change with respect to each other. To evaluate and compare performance, we need a more comprehensive evaluation metric, therefore we use both recall and precision.

Finally, we discuss the distinction between annotation and retrieval performance. Annotation performance reflects the quality of annotations by themselves and it matters most when the application involves actually displaying the annotations to the user. For example, if we are building an image browsing tool, it is advantageous to display images paired with their respective annotation. Text can represent visual information very concisely and in general people find it helpful when images are accompanied by captions or some other kind of explanatory information. For such an application, we would want each annotation to contain as many correct words as possible.

A different kind of application is an image retrieval system where we

need annotations only as image proxies. The internal representation of retrieved documents is not displayed, so the user would not see the annotations themselves. Therefore, annotations do not have to be lists of specific words. In fact, it would be more advantageous if rather than a few fixed words, the annotation of an image is a probability vector, whose components reflect the probability of associating the corresponding word with the image. To understand why this is more helpful than assigning concrete words, recall the Boolean and Vector Space models discussed in Section 2.4.

When given annotations made up of actual words, we can think of each word in the vocabulary as being either selected or not selected for a particular annotation. Therefore, we can represent such annotations as binary vectors where we assign 1 if the annotation contains the corresponding word and 0 otherwise. These binary vectors would have the same dimension as probability vectors, with one component for each word in the vocabulary. Therefore, with annotations, which give words explicitly (this situation corresponds to the Boolean model), the system would only be able to return a set of relevant images without a notion of ordering in terms of relevance. With annotations that give a probability for each word (this situation corresponds to the Vector Space model), the system would be able to rank images with respect to their degree of relevance. Typically, the second scenario is preferable in IR - otherwise the user would be compelled to go through the entire retrieved set to convince herself that she would not miss an appropriate image. On the other hand, if the results are ranked, she would have the best matches right at the top of the list.

Probabilistic annotations are directly generated by our statistical models; they are constructed from the $P(w, I)$ values, which reflect how likely it is to annotate the image I with the word w. Then for a multiple-word

query $Q = q_1, ..., q_k$ and an image $I = v_1, ..., v_m$ we can use a general language modeling technique to compute $P(Q|I)$, the probability of the model of I generating the query Q:

$$P(Q|I) = \prod_{j=1}^{k} P(q_j|I)$$

Although probabilistic annotation allows us to formulate queries of arbitrary length, in our experiments we evaluate retrieval performance using only single-word queries, one for each word in the controlled vocabulary, averaging recall and precision across the queries.

Fixed-length annotations are created by sorting the word probabilities in decreasing order and picking the top n words for the actual annotation. We must decide in advance how many words to select for an annotation, keeping in mind that annotation length directly influences performance. Shorter annotations mean higher precision as we assign few but good words; longer annotations mean higher recall as we assign many (both correct and incorrect) words. Notice that we cannot create annotations of varying length because all images have 24 visterms and therefore nothing in the visual representation indicates how long the true annotation is. Thus, an additional shortcoming of explicit annotations for ranked retrieval is that length normalization would have absolutely no effect.

Because we propose two very general models, which can be applied in various contexts, we are equally interested in how incorporating clustering information affects annotation performance and retrieval performance. Therefore, when presenting experimental results we will report both alternatives - retrieval performance using probabilistic annotations and annotation performance using fixed-length annotations.

And finally, we want to make the following clarification. We use a retrieval technique, relevance modeling, to automatically annotate images,

the query being the image we want to annotate and the document collection being the set of training images. Then we use a different retrieval technique, language modeling, to retrieve images, the query being a word and the document collection being the set of test images, already automatically annotated.

Chapter 6

Experiments and Results

In this chapter we lay out the experiments conducted to test our hypothesis that information shared by similar images can be effectively employed to improve the performance of an image retrieval system. First, we present procedures and results, and then we analyze our findings.

6.1 Baseline Model

To measure the improvement gained from incorporating clustering into the Cross-media Relevance model (CMRM), we compare the performance of our cluster-based approach with that of the original method which is based exclusively on individual image ranking. The baseline model is introduced briefly in Section 3.4 and discussed in depth in Section 4.1.

The results obtained by Lavrenko *et al*, who first proposed the Cross-media Relevance model, are reported in [13]. Although we use the same dataset that they used to train and test CMRM, we cannot directly compare our results with theirs: they apply a segmentation algorithm to divide the images into blobs while we use visterm representations generated from a rectangular-grid partitioning. One immediate difference is that segmentation produces different number of blobs per image while partitioning

produces the same number of visterms - in our case 24 per image. So to make a fair comparison between cluster-based and image-based automatic annotation and retrieval, we first train CMRM with visterms instead of blobs.[1]

Using Jelinek-Mercer smoothing, we find that the optimal value of α, the parameter for smoothing words in the image models, is equal to the one reported by Lavrenko *et al*, $\alpha = 0.1$. This is not surprising. Note that the textual representation of training images (their annotations) do not change as a result of modifying the visual representation. In fact, the two representations are generated completely independently: annotations by manual labeling and visual tokens by automatic image processing. Also, when the training data is analyzed, word and visterm distributions are computed separately of each other, as Equations 4.1 and 4.2 indicate. When we use the same example images for training, even though represented with visterms instead of blobs, we should still get exactly the same probability distributions $P(w|J)$ for $w \in V$, and therefore the same α.

On the other hand, in our experiments the optimal value for β, the parameter for smoothing visterms in the image models, turns out to be $\beta = 0.7$ while Lavrenko *et al* report $\beta = 0.9$. The disparity is likely due to differences between blobs and visterms arising implicitly from the fact that they are generated according to different procedures. We have decided to use visterms because studies show that visterms are as effective as blobs while they require much less time and computational resources to generate [14].

An interesting question to consider is why the value for α is low while the value for β is high. Recall that the value of a smoothing parameter sig-

[1]The preliminary image processing work required to generate the visterms was completed at the Multimedia Retrieval Lab, Center for Intelligent Information Retrieval, University of Massachusetts, Amherst.

nifies how much the maximum likelihood estimation of the specific model is interpolated with a general background model. The smaller the parameter is, the more weight is put on the specific model, which implies that we "trust" it more. In theory, when we model a process, it is reasonable to be more confident in our estimations if we use hundreds of observations rather than tens of observations.

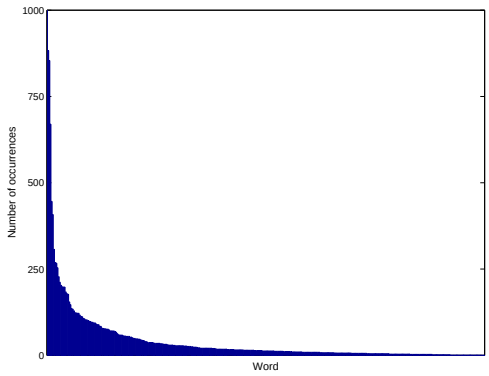


Figure 6.1: Distribution of individual words where words are sorted in decreasing order with respect to number of occurrences. Few words occur very frequently and most words occur only rarely. This distribution follows Zipf's Law which says that the inverse relationship between the frequency f of a word and its position in the sorted order of words - its rank r, has the form $f \times r = k$ where k is a constant.

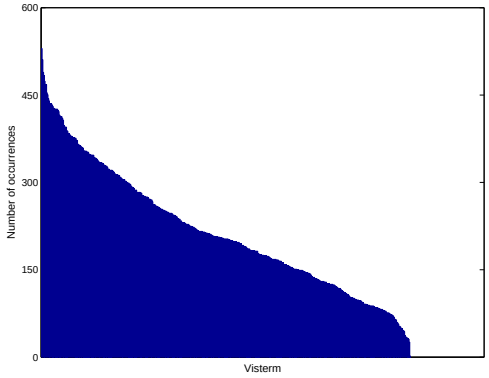


Figure 6.2: Distribution of individual visterms where visterms are sorted in decreasing order with respect to number of occurrences. Most visterms occur between 100 and 300 times.

The quality of a language model is measured by its power to discriminate between hypotheses. For example, in speech recognition the hypotheses are all the possible transcriptions of a word the speaker utters; in machine translation the hypotheses are all the possible translations of a word in the source language. In our particular case the hypotheses are all the possible words that we can assign to an image. Why visterms need more smoothing than words and therefore have less power to distinguish between

hypotheses, is explained by the fact that words and visterms are distributed differently across the collection. As shown in Figure 6.1 and 6.1, words have a very skewed distribution while visterms have a much more uniform distribution. Intuitively, the more uniform a distribution is, the less power it has to discriminate.

For completeness, we also test using Bayesian smoothing with Dirichlet prior instead of Jelinek-Mercer smoothing, since the former technique is usually better for full-text retrieval. We optimize β for Bayesian smoothing by setting α to its best value 0.1 and varying β from 10 to 80 in 10-size increments. (Recall from Section 4.1.1 that in this case the smoothing parameter plays the role of a pseudo count added to the actual number of occurrences, so there is no reason to restrict the values in the range [0,1].) We get $\beta_{max} = 40$. The result of comparing the two different methods of smoothing is presented in Figure 6.3.

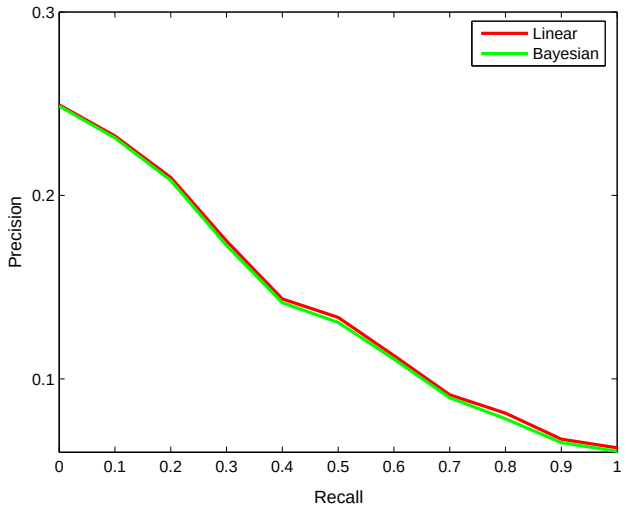


Figure 6.3: Retrieval performance of CMRM using two different techniques for smoothing visterms. (*Linear* refers to Jelinek-Mercer smoothing; the alternative name comes from the fact that this technique performs a linear interpolation between the models of seen and unseen events.)

The experiment shows that the performance is virtually the same no matter whether we use Bayesian or Jelinek-Mercer smoothing. The expla-

nation lies in the conceptual differences between Bayesian and Jelinek-Mercer smoothing. As discussed in Section 4.1.1, Bayesian smoothing yields smaller α_D for longer documents, hence it smooths longer documents less - a reasonable distinction given the fact that longer documents provide more observations from which to estimate the distribution. However, in our collection images have representations of the same length - 1-5 words and exactly 24 visterms. This implies that the smoothing parameter α_D is the same for all document, just like with Jelinek-Mercer smoothing. Therefore, in this particular case there is no actual difference between the two smoothing techniques.

To implement cluster-based annotation and retrieval, we first need to partition the set of training images into subsets. Of course, this has to be completed automatically or otherwise we would compromise one major requirement - to build a system that works entirely automatically or at least with minimal manual intervention. Therefore, we apply an unsupervised clustering algorithm (one of the four approaches described in Section 2.8: K-means, single linkage, complete linkage or group-average). Another characteristic of the generated clusters is that they are global because we divide the images into groups prior to performing any other kind of image analysis which might be specific to the image we want to annotate. Thus, clusters are created with respect to the global similarity structure of the collection rather than with respect to a particular image or feature.

6.2 CQL - Ranking Clusters

The goal of this set of experiments is to examine whether clusters are as effective as individual images for estimating the joint distribution of words and visterms.

We start by partitioning the training images into clusters and constructing textual and visual representations of the clusters. Recall that we consider images to be "bags" of visterms, and annotations - "bags" of words. A bag is a set with repetitions: an unordered group of elements where an element can occur more than once. This definition of an image and its annotation allows us to combine images and their annotations without loss of information. The rest of the procedure is described in length in Section 4.2 and is summarized below:

1. Represent training images in terms of count vectors, indicating number of occurrences for both words and visterms.

2. Weight the count vectors using Latent Semantic Indexing, reducing dimensionality to 100.

3. Cluster the weighted vectors to find groups of similar images.

4. Construct clusters, combining the words and visterms of comprising images.

5. Build probabilistic models of cluster frequencies, smoothing with collection frequencies.

6. Given an image I to annotate, rank clusters in terms of their visual similarity with I. Taking into account what words tend to co-occur with the visterms of similar clusters, compute probabilities of words being selected to annotate I.

Before we can evaluate this model, we need to perform a set of initial experiments to optimize its parameters - γ, δ and θ. (Recall that according to the definitions in Section 5.3 γ determines the level of smoothing for words in cluster models, δ determines the level of smoothing for visterms

in cluster models, and θ determines the desired number of clusters if we use K-means or the similarity threshold if we use an agglomerative technique.)

First we set the smoothing parameters γ and δ choosing somewhat arbitrarily to cluster the image space using K-means with $\theta = 100$. Following the procedure described in Section 5.3, we start by setting $\delta = 0.5$ and $\theta = 100$ and varying γ from 0.1 to 0.9. After cross-validating we get $\gamma_{max} = 0.1$. Next, we fix $\gamma = 0.1$ and $\theta = 100$ and vary δ. We get $\delta_{max} = 0.1$. Finally, we fix $\gamma = 0.1$ and $\delta = 0.1$ and vary θ from 50 to 300 in 25-size increments. We get $\theta_{max} = 250$.

To summarize, we get the following optimal values for the two smoothing parameters when we use Jelinek-Mercer smoothing: $\gamma_{max} = 0.1, \delta_{max} = 0.1$. Substituting with the actual values in Equations 4.6 and 4.7, we get the following linear interpolation between cluster and collection frequencies:

$$\begin{aligned} P(w|G) &= 0.9\frac{\#(w, G_J)}{|G_J|} + 0.1\frac{\#(w, T)}{|T|} \\ P(v|G) &= 0.9\frac{\#(v, G_J)}{|G_J|} + 0.1\frac{\#(v, T)}{|T|} \end{aligned}$$

6.2.1 Jelinek-Mercer vs. Bayesian Smoothing

While image representations have the same length, images are not equally distributed among clusters. When $\theta = 250$ cluster sizes vary from 1 to 71 images with average of 18. Therefore cluster representations have different lengths and a reasonable question to ask is whether Bayesian smoothing with Dirichlet will perform better than Jelinek-Mercer smoothing.

Since there are two smoothing techniques we can use and we smooth words and visterms separately, there are four different combinations: Jelinek-Mercer for both words and visterms, Bayesian for visterms and Jelinek-Mercer for words, Bayesian for words and Jelinek-Mercer for visterms, and

Bayesian for both words and visterms. We want to evaluate each combination but we do not need to run the cross-validation procedure four times as we already know the best parameters for Jelinek-Mercer smoothing and for the clustering algorithm. First, we optimize γ for Bayesian smoothing by setting $\delta = 0.1$ and $\theta = 100$ and varying γ from 10 to 70 in 10-size increments. We get $\gamma_{max} = 10$. Next, we optimize δ for Bayesian smoothing by setting $\gamma = 0.1$ and $\theta = 100$ and varying γ from 10 to 80 in 10-size increments. We get $\delta_{max} = 50$.

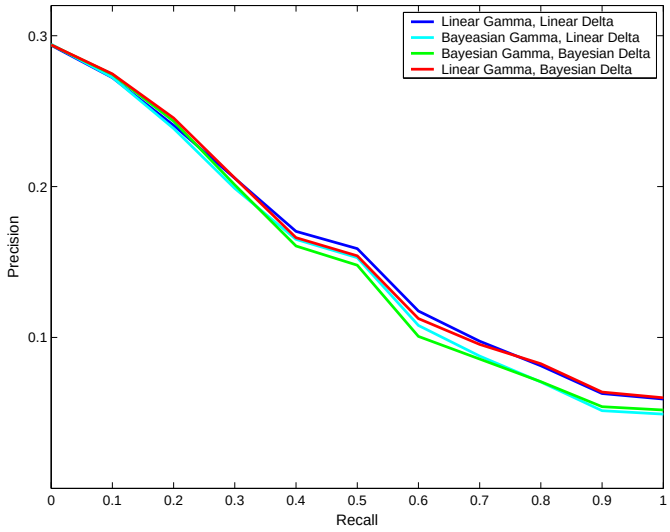


Figure 6.4: Retrieval performance of CQL using two different smoothing techniques for smoothing words and visterms.

The result of comparing the different methods of smoothing is presented in Figure 6.4. It shows that none of the techniques outperforms the rest. Our intuition is that the variance in cluster sizes is not enough for length normalization to have a significant effect. Also, notice that smoothing does not really affect performance at high precision because it is more effective when there are many nonrelevant documents: it is when we are less confident about the foreground frequencies that we start relying more on the background frequencies. If a word appears several times in a cluster we can reasonably assume that is it semantically important. On the other

hand, if a word appears only once, then we should take into account its overall frequency to decide whether it is semantically important or not.

For the final evaluation of CQL with K-means clustering as well as for the rest of our experiments on CQL, we use Jelinek-Mercer smoothing for both words and visterms setting $\gamma = \gamma_{max} = 0.1$ and $\delta = \delta_{max} = 0.1$.

6.2.2 K-means vs. Agglomerative Clustering

To investigate whether the model is robust in terms of the algorithm used to find groups of similar images, we run another set of experiments, this time changing the clustering algorithms and adjusting the clustering parameter.

In the experiments so far we used K-means clustering. As discussed in Section 2.8.3, K-means defines a cluster in terms of its centroid, the center of mass of the comprising member elements, and it requires specifying the number of clusters into which to group the elements as an input argument. Not having any prior knowledge about the right value of this parameter, we have to apply the algorithm for different values and use the one which gives the highest F-measure for the final evaluation.

An alternative to K-means is agglomerative clustering, discussed in Section 2.8.2, which starts by assigning each object to its own cluster and then proceeds iteratively by merging the two closest clusters until there is only one cluster left. We introduced three variants of this approach that differ only in the definition of the similarity function, which determines how the distances between clusters are computed. Single-linkage takes the distance between the pair of closest elements, complete-linkage takes the distance between the pair of most distant elements, and group-average takes the average distance between pairs.

We train all three techniques using Jelinek-Mercer smoothing for the words and visterms with γ and δ set to their optimal values, $\gamma_{max} = 0.1$ and

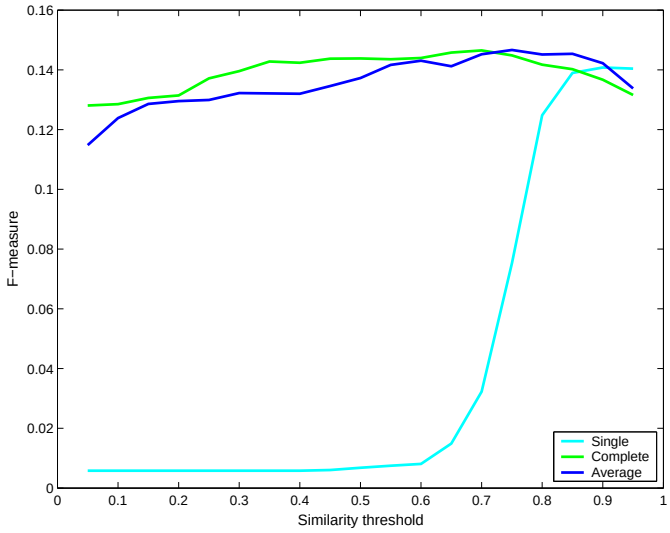


Figure 6.5: Comparison of the performance of three agglomerative clustering algorithms with CQL.

$\delta_{max} = 0.1$. As already discussed, agglomerative clustering algorithms take as input a similarity threshold θ which determines when to stop merging clusters. Since similarity is a real number in the range [0,1] we vary θ from 0.05 to 0.95 in 0.05-value increments. Results are reported in Figure 6.5.

Complete-linkage and group-average have comparable performance and both consistently outperform single-linkage for all but the highest threshold levels. Recall from Section 2.8 that the primary difference between complete-linkage and group-average, on one hand, and single-linkage, on the other hand, is the compactness of generated clusters. Single-linkage is very sensitive to spread-out data and we can see that for values of θ less than 0.85, its performance quickly deteriorates. The other two methods are much more robust and their performance decreases only very slightly as we continue to augment clusters.

Although it is somewhat difficult to see, the highest point in the graph corresponds to group-average at $\theta = 0.75$ and this is the value that we use for the final experiment comparing K-means and agglomerative clustering. The result of this experiment is shown in Figure 6.6.

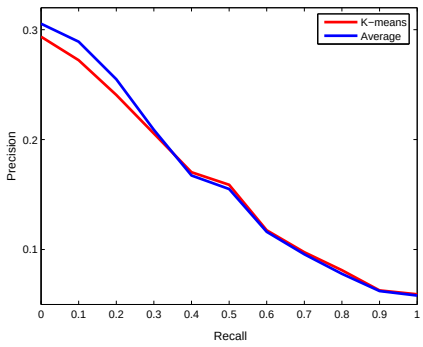


Figure 6.6: Comparison of K-means and group-average clustering for image retrieval based on ranking clusters of similar images.

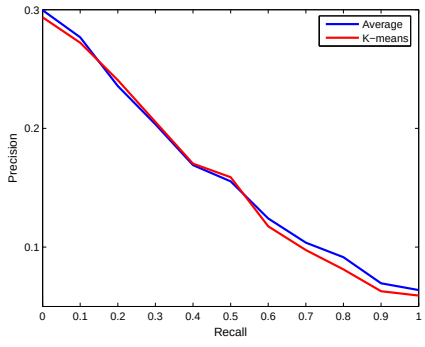


Figure 6.7: Comparison of K-means and group-average when both clustering algorithms were used to produce 250 clusters.

Group-average clustering at $\theta = 0.75$ generates 984 clusters with average size of 4.6 images. Because the similarity threshold is relatively high, group-average produces small but precise clusters - all cluster members are relatively close to each other and to the cluster centroid. In terms of their characteristics, group-average clusters are in between the individual images and the bigger and looser K-means clusters. Therefore, they are more consistent in terms of membership variability. Recall that K-means is sensitive to outliers and because it has to generate exactly 250 clusters, it might have forced together elements that are relatively dissimilar. And you can see from Figure 6.6 that the difference in performance is at high recall. Intuitively, reducing the compactness of clusters by including images that do not fit so well with the other cluster members reduces precision because we inevitably include some incorrect information.

In order to show that the difference between K-means and group-average is not simply in the number of clusters but in the quality of clusters, we use group-average to produce exactly 250 clusters and compare it with K-means again. (Recall from Section 2.8.2 that we can generate an exact number of clusters by cutting the dendrogram at an arbitrary level k producing $n-k+1$ clusters.) The result is reported in Figure 6.7. Because we cut the

dendrogram much higher, we compromise our criterion for including images in the same cluster and hence we lose the advantage that agglomerative clustering has for high precision. But performance is now better at high recall, where the influence of outliers on K-means starts growing. Note that this is evidence for the implicit tradeoff between recall and precision. By including more information in a cluster, we reduce its focus but increase its scope.

After we run experiments with various smoothing techniques and clustering algorithms, we conclude that CQL, the cluster-based model which ranks clusters instead of images, has the best performance with Jelinek-Mercer smoothing at $\gamma = 0.1$ and $\delta = 0.1$ and group-average clustering with $\theta = 0.75$. Now that we have trained the model we can compare it with CMRM, the original relevance modeling approach which ranks images. As you can see from Figure 6.8, CQL substantially outperforms CMRM on the retrieval task, except for precision at high recall.

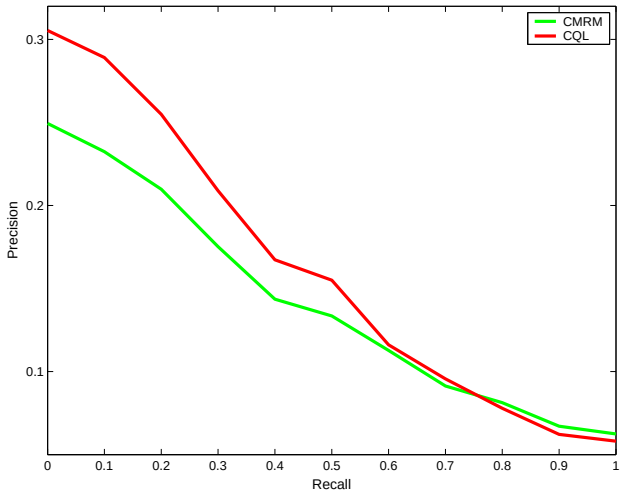


Figure 6.8: Performance of CQL and CMRM on the ranked retrieval task. The improvement of our cluster-based approach is statistically significant with p-value 0.001045 according to the Wilcoxon signed rank test.

Before we analyze this result, we want to remind readers how a recall-

precision curve is drawn. We consecutively issue 260 1-word queries and for each one we know which the relevant images are by looking at the manual annotations.[2] To evaluate the result of a particular query, we start at the top of the ranked list, checking whether an image is relevant or not, and move down the list until we have found 10% of the actually relevant images. We stop and measure precision up to this point of recall, and continue the procedure for all points of recall that we are interested in - 10% through 100%. Finally we average across the 260 queries. Therefore, the left portion of the curve corresponds to highly ranked images, and as we move to the right we add more and more images that we are not so confident about.

It is obvious from Figure 6.8 that the significant improvement of using clusters is in precision and not recall. In fact, our hypothesis that clusters should help to compensate for information missing in the very short image representation, implies that CQL is better at assigning rare words. Intuitively, missing information hurts precision because we fail to notice potentially good co-occurrences. Missing co-occurrences for frequent words, for which there are many other examples anyway, is relatively less important. The fact that CQL is better at annotating rare words is evident in Table 6.1 which shows that CQL assigns 15 more words correctly at least one time.

6.3 CBDM - Smoothing with Clusters

The goal of this set of experiments is to examine whether clusters are as effective as the collection for smoothing the models of individual images.

[2]There are a total of 371 words in the vocabulary. Many words appear only once, so when the complete set of images is divided into subsets for training and testing, those singletons appear once in either the test images or training images. For the final evaluation we take the intersection of training words and test words, thus obtaining 260 words for which we can both train the system and test it.

	CMRM	CQL
Average annotation accuracy	28.35%	**35.10%** (0.000005)
Nonzero words	86	**101**

Table 6.1: Performance of CQL and CMRM on the annotation task. The numbers in brackets report p-values for the Wilcoxon signed rank test. Improvement is statistically significant if the p-value is less than 0.05. Nonzero words are words that have been correctly assigned to at least one image, so they have nonzero recall and precision.

We have already created the clusters when we implemented the previous technique; now we are going to incorporate them in the Cross-media Relevance model in a different way. We do not discuss the procedure here, since it is described in length in Section 4.3, but we give a brief summary below:

1. Represent training images in terms of count vectors, indicating number of occurrences for both words and visterms.

2. Weight the count vectors using Latent Semantic Indexing, reducing dimensionality to 100.

3. Cluster the weighted vectors to find groups of similar images.

4. Construct clusters, combining the words and visterms of comprising images.

5. Build probabilistic models of cluster frequencies, smoothing with collection frequencies. Then, build probabilistic models of image frequencies, smoothing with cluster frequencies.

6. Given an image I to annotate, rank images in terms of their visual similarity with I. Taking into account what words tend to co-occur with the visterms of similar images, compute probabilities of words being selected to annotate I.

Therefore, instead of considering each cluster as a big pseudo image, we use clusters as generalizations of their comprising elements. However, notice that this is not enough to guarantee nonzero probabilities for all words and visterms. For example, a term which appears only once in the collection is assigned to one particular cluster; for the rest of the clusters its frequency is zero. Therefore, the language models of clusters need to be smoothed too, and for this we use the collection. For both words and visterms, the two-step smoothing procedure is a linear interpolation between three distributions. The exact definitions are given in Equations 4.8 and 4.9.

An immediate consequence of using a more complex statistical model is the need to optimize five instead of three parameters - $\alpha, \beta, \gamma, \delta$ and θ. (We remind that according to the definitions in Section 5.3, α determines the level of smoothing for words in image models and β determines the level of smoothing for visterms in image models. γ, δ and θ have the same definition as in the previous section.) This means that we have to run the cross-validation procedure five consecutive times but at least we already have generated the clusters.

As with CQL, in the first round of experiments we set the smoothing parameters $\alpha, \beta, \gamma, \delta$. To be consistent, we use the K-means clusters generated with $\theta = 100$ and Jelinek-Mercer smoothing for all parameters.

Following the procedure described in Section 5.3, we start by setting $\beta = 0.5, \gamma = 0.5, \delta = 0.5$ and $\theta = 100$ and varying α from 0.1 to 0.9. After cross-validating we get $\alpha_{max} = 0.1$. Next, we fix $\alpha = 0.1$, keep $\gamma = 0.5, \delta = 0.5, \theta = 100$ and vary β from 0.1 to 0.9. We get $\beta_{max} = 0.6$. Next, we fix $\alpha = 0.1, \beta = 0.6, \delta = 0.5, \theta = 100$ and vary γ from 0.1 to 0.9. We get $\gamma_{max} = 0.7$. Finally, we fix $\alpha = 0.1, \beta = 0.6, \gamma = 0.7$ and $\theta = 100$ and vary δ. We get $\delta_{max} = 0.1$.

To summarize, we get the following optimal values for the four smoothing parameters when we use Jelinek-Mercer smoothing: $\alpha_{max} = 0.1, \beta_{max} = 0.6, \gamma_{max} = 0.7, \delta_{max} = 0.1$. Substituting with the actual values in Equations 4.8 and 4.9, we get the following linear interpolation between image, cluster and collection frequencies:

$$\begin{aligned} P(w|J) &= 0.9\frac{\#(w,J)}{|J|} + 0.1\left(0.3\frac{\#(w,G_J)}{|G_J|} + 0.7\frac{\#(w,T)}{|T|}\right) \\ &= 0.9\frac{\#(w,J)}{|J|} + 0.03\frac{\#(w,G_J)}{|G_J|} + 0.07\frac{\#(w,T)}{|T|} \\ P(v|J) &= 0.4\frac{\#(v,J)}{|J|} + 0.6\left(0.9\frac{\#(v,G_J)}{|G_J|} + 0.1\frac{\#(v,T)}{|T|}\right) \\ &= 0.4\frac{\#(v,J)}{|J|} + 0.54\frac{\#(v,G_J)}{|G_J|} + 0.06\frac{\#(v,T)}{|T|} \end{aligned}$$

Of course, we want to consider these results in the context of our previous findings for CMRM and CQL. To begin with, the values of the image parameters α and β are very similar to what is optimal for CMRM, 0.1 and 0.7 for α and β respectively, and this is very reasonable. We use the same image representations and therefore the informativeness of images has not changed. Specifically, a lot of weight is given to word image frequencies. The explanation, just as with CMRM in Section 6.1, is the Zipfian distribution of words. Most of the words occur very few times and if we smooth out their actual occurrences, then the model would exhibit a strong tendency to assign frequent words. If we rely almost exclusively on the overall frequencies, then annotation would be strikingly similar to each other as we assign 'sky', 'water, 'grass' and few other very frequent words to most images. While these words might be correct (recall that the dataset is a collection of wildlife photographs), they describe background rather than foreground objects and hence we could consider them less semantically important than

rare words like 'tiger'.

On the other hand, the value of γ is different from what we found optimal for CQL. The difference is likely due to the fact that clusters play very different roles in the two cluster-based models - in CQL we directly compare clusters with the image to be annotated, while in CBDM we smooth training images with clusters. In the first case, we want to build cluster models which are good for discriminating because we rank the clusters themselves; in the second case, we want to build cluster models which are good for capturing global similarity patterns because we generalize with clusters.

Finally, we fix the smoothing parameters at their optimal values and vary θ from 50 to 300 in 25-size increments to find the optimal value for the K-means clustering parameter. Again, we get $\theta_{max} = 250$.

6.3.1 Jelinek-Mercer vs. Bayesian Smoothing

Next, we determine what effect Bayesian smoothing has on performance. Recall from Section 6.1 that for CMRM it acts just like Jelinek-Mercer smoothing, the reason being that images have the same length. Here too, there is no reason to smooth images with the Bayesian approach. But since clusters have different lengths, Bayesian smoothing can turn out to work better for clusters.

We discover a similar pattern as in CQL: Bayesian smoothing performs slightly worse than Jelinek-Mercer for either words or visterms, and the errors are compounded if we use Bayesian smoothing for both. Again, we point out that the influence of the smoothing technique is particularly evident at high recall. As we retrieve more and more images, we need to choose from images that are less confidently associated with the query word and we rely more and more on the fallback method in our estimations.

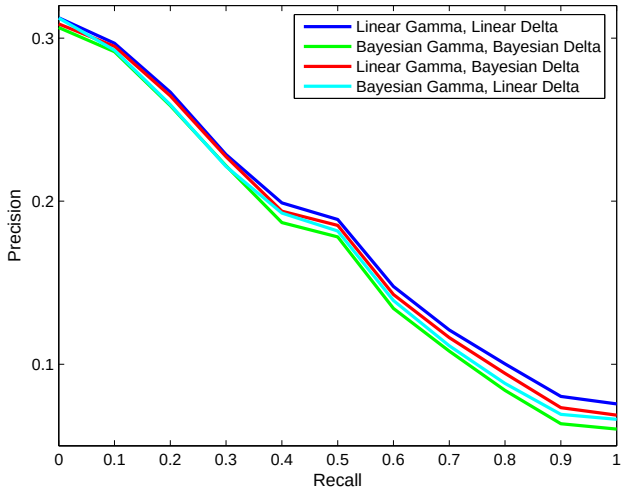


Figure 6.9: Retrieval performance of CBDM using two different smoothing techniques for smoothing words and visterms.

6.3.2 K-means vs. Agglomerative Clustering

To investigate whether the model is robust in terms of the algorithm we use to find groups of similar images, we run another set of experiments using in turn one of the three agglomerative clustering approaches: single-linkage, complete linkage and group-average.

We train all three techniques using Jelinek-Mercer smoothing for the words and visterms with α, β, γ and δ set to their optimal values, $\alpha_{max} = 0.1, \beta_{max} = 0.6, \gamma_{max} = 0.7$ and $\delta_{max} = 0.1$. We vary the similarity threshold θ from 0.05 to 0.95 in 0.05-value increments. The results are reported in Figure 6.10.

As a whole, the curves are very similar to what we get for CQL: complete-linkage and group-average have comparable performance and both consistently outperform single-linkage for all but the highest threshold levels. The performance of single-linkage, which is very sensitive to spread-out data, quickly deteriorates for values of θ less than 0.85. The other two methods are much more robust. However, notice that the peak of complete-linkage and group-average has shifted to the left when compared

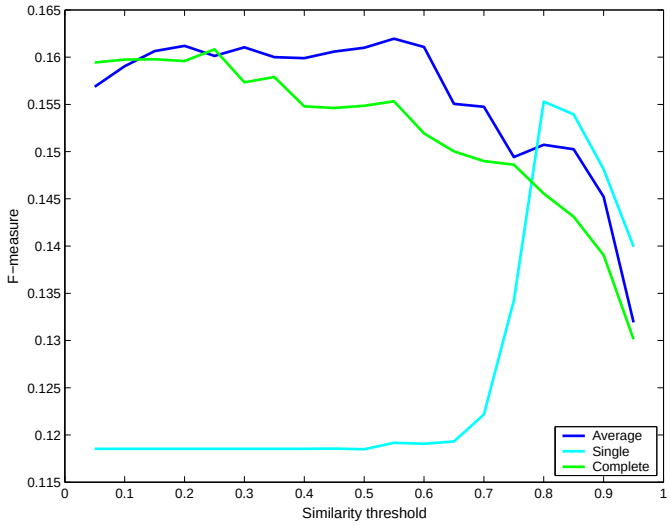


Figure 6.10: Comparison of the performance of three agglomerative clustering algorithms with CBDM.

with results in Figure 6.5: for complete-linkage it is at $\theta = 0.25$ and for group-average it is roughly in the middle at $\theta = 0.55$. Therefore, the best performing agglomerative clusters in this case are bigger than for CQL because we cut the dendrogram at a lower similarity threshold and therefore we continue to merge clusters even as dissimilarity between merged clusters continues to grow. In fact, performance would be low if we stop merging clusters too soon. The reason why bigger clusters work better in CBDM is that we use them for making generalizations of patterns observed in individual images. If the clusters are too small, then our generalizations are not useful enough as they fail to capture existing global structures. In this case we care less about the compactness of the clusters themselves, and can allow some relatively dissimilar elements to join the same cluster.

The highest point in Figure 6.10 corresponds to group-average at $\theta = 0.55$ and this is the value we use for the final experiment comparing K-means and agglomerative clustering. The result of this experiment is shown in Figure 6.11.

The performance of the two clustering algorithms is similar and K-

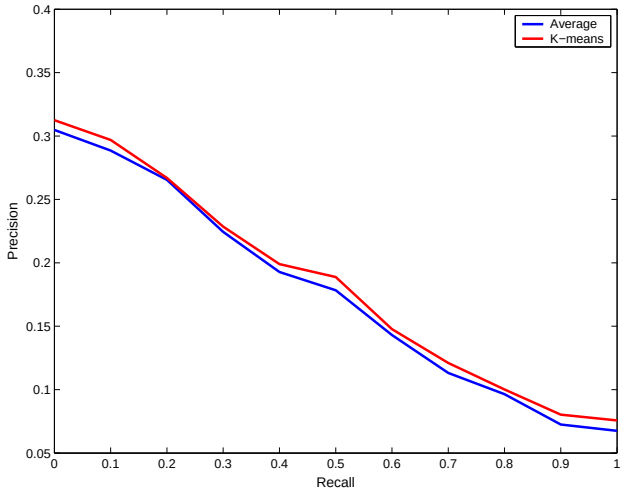


Figure 6.11: Comparison of K-means and group-average agglomerative clustering for image retrieval based on smoothing images with clusters.

means is only marginally better. Our intuition is that the shape and quality of clusters matter less in this case because we do not use the clusters for direct comparison with the image to be annotated. Therefore, a general-purpose algorithm like K-means works comparably well as a conceptually more sophisticated algorithm like group-average.

After we run experiments with various smoothing techniques and clustering algorithms, we conclude that CBDM, the cluster-based model which smooths image frequencies with cluster frequencies, has the best performance with Jelinek-Mercer smoothing at $\alpha = 0.1, \beta = 0.6, \gamma = 0.7$ and $\delta = 0.1$ and K-means clustering with $\theta = 250$. Now that we have trained the model we can compare it with CMRM, the original relevance modeling approach which ranks images. As you can see from Figure 6.12 and Table 6.2, CBDM substantially outperforms CMRM on both the retrieval and annotation tasks. The performance of CQL is shown for comparison.

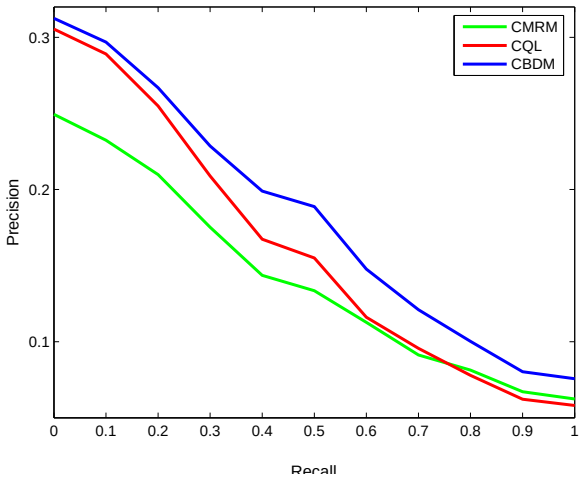


Figure 6.12: Performance of CBDM, CQL and CMRM on the ranked retrieval task. The improvement of the cluster-based approaches is statistically significant with p-value = 0.001045 for CQL and p-value = 0.000003 for CQL according to the Wilcoxon signed rank test.

	CMRM	CQL	CBDM
Average annotation accuracy	28.35%	35.10% (0.000005)	**37.75%** (0.000000)
Nonzero words	86	101	**102**

Table 6.2: Performance of CMRM, CQL and CBDM on the annotation task.

6.4 Discussion

In their study Liu *et al* report that text-retrieval CQL sometimes performs slightly better and sometimes slightly worse than document-based retrieval and they conclude that in general it is just as effective [16]. For text-retrieval CBDM they report a statistically significant improvement over the baseline model. Our results show that both cluster-based methods are better with statistical significance than CMRM. Of course, we evaluate the proposed techniques on one collection only, so to confirm that CQL and CBDM are better in general than CMRM for automatic annotation and retrieval we should conduct experiments on other collections in addition to the Corel dataset. (Liu *et al* test on five different document collections.)

It is essential to keep in mind that the relevance modeling techniques

described in the previous sections, both the cluster-based and the image-based, adapt a text retrieval approach to annotate images. However, full-text documents and images have different features. Most importantly, with their 1 to 5 words and 24 visterms image representations are substantially shorter than written documents. For example, the average size of documents in the five collections used by Liu *et al* varies between 412 and 874 terms.

The significant improvement when using clusters instead of images is evident in the increase of both recall and precision. A more subtle proof of our hypothesis is what parameters turn out to work best for smoothing clusters in CQL. Since $\gamma = 0.1$ and $\delta = 0.1$, both words and visterms require minimal smoothing with the collection frequency. This implies that more weight is given to the observed occurrences in clusters rather than the background frequencies, therefore we can conclude that $P_{MLE}(\cdot|C)$ from clusters is closer to the true distribution than $P_{MLE}(\cdot|J)$ from images alone, particularly for visterms.

Our hypothesis is based on the assumption that the underlying distributions of similar images are themselves similar but this should not imply that images clustered together have relatively the same probabilities for all words and visterms. Actually, we would expect images that are semantically similar to be similar in some important aspects of their visual appearance and yet be dissimilar in other less important aspects. For example, foreground objects are usually more relevant than background objects because users would consider them more interesting. However, by representing images as bags of visterms we discard any implicit positional information; consequently the model has no way of distinguishing between foreground and background visterms, and therefore no way of distinguishing between appearance and semantics - the very capability that makes

humans good annotators.

By using clusters to estimate probability distributions we reinforce the weight of some terms - the terms that co-occur in the images clustered together. We believe that performance improves because the reinforced terms are semantically important, thus we achieve a simple but effective image analysis. Clustering allows us to take advantage of this assumption in a natural way by finding groups of similar images. The actual observations from each individual distribution on which we base our approximation are limited by the number of words and visterms per image, but we can complement this insufficient information with information extracted from similar images, thus getting a closer, more reliable approximation of the underlying distribution.

To illustrate our claim that clusters are informative, we look at the performance of CQL obtained by ranking only a few selected clusters, the criterion for selection being the similarity between the test image I and the cluster G. To do this, we first need to compute this similarity - we use the value $P(I|G)$, which measures the probability of G generating I. This is LM ranking with the query being I and the document being G. The specific formula is:

$$P(I|G) = P(v_1, ..., v_m|G) = \prod_{i=1}^{24} P(v_i|G)$$

Figure 6.13 shows the performance we get with only the 1, 5 or 10 highest ranked clusters out of 250. While one cluster, even though it is the very best one, is not enough to assign good annotations, the top 5 clusters give virtually the same performance that we get using all of the clusters. Using additional clusters only marginally increase performance at high recall but not precision because the most relevant information is contained in the

top clusters. The fact that only 5 examples are sufficient to find suitable annotation words shows that clusters have substantial information content.

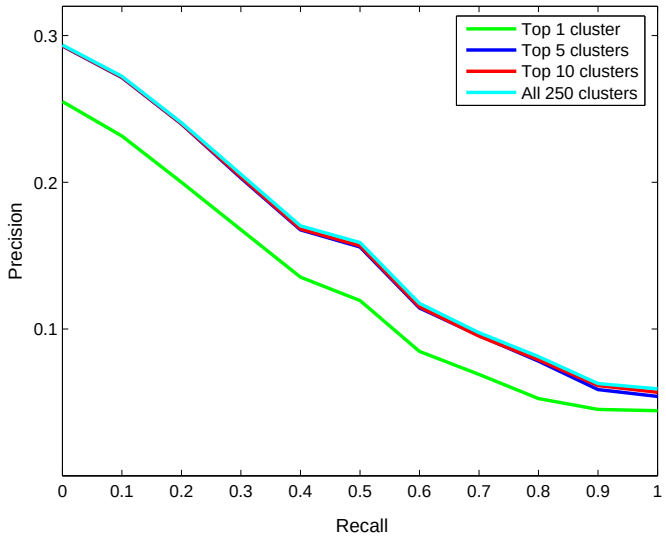


Figure 6.13: Informativeness of clusters: We get very similar performance using the 5 visually most similar clusters and all of the 250 clusters generated by K-means.

Another point to take into account is that after clustering we have fewer examples for comparison on which to base probability estimation. Consider that we group 4500 training images into 250 clusters, therefore we have 18 times fewer examples and yet we get comparable performance. This is another convincing proof the clusters are meaningful and they provide useful and discriminating information.

In this line of reasoning, a straightforward advantage of CBDM over CQL is that we have more training examples - as many individual images as we are provided with. And more examples implicitly mean that, for a given query word, we discover more related visterms, which increases both recall and precision.

The choice of a clustering technique also has a significant effect on the final performance. We made the important observation that agglomerative clustering has better performance than non-hierarchical clustering in CQL

and similar performance in CBDM. Since with the agglomerative approach we have more control on the quality of clusters rather than on their number, this could be extremely useful when working with huge collections - in such cases setting parameters experimentally by varying their value in small increments can be prohibitive. It is more realistic to learn how to set the similarity threshold automatically by taking into consideration the specific properties of images, rather than analyzing the global similarity structure of a collection (which can moreover change over time by adding more images) and trying to guess the number of clusters that images fall naturally into.

Agglomerative clustering is also more powerful in terms of providing us with an opportunity to control the coherence of clusters. From our analysis so far it is clear that we propose to use clusters into two conceptually different ways - on one hand, as training examples which are somewhat more general than images, and on the other hand as background collections which are somewhat more specific than the entire collection. To visualize this idea, imagine images, clusters and the collection appearing in this order from left to right on an x-axis which shows how the granularity of document representations varies from very specific to very general. The actual position of clusters along the axis can change and depends on their compactness - tighter clusters are closer to images, looser clusters are closer to the collection. The ability to determine the quality of clusters is very important and with agglomerative clustering we can bring up the similarity threshold to get very compact clusters, which works well for CQL, or bring it down to get more general clusters, which works well for CBDM.

Chapter 7

Conclusion

In this thesis we proposed two cluster-based techniques for automatic image annotation, Cluster Query Likelihood (CQL) and Cluster-based Document model (CBDM). Our main contribution was to develop and evaluate a method for incorporating cluster statistics into a general Cross-media Relevance model (CMRM), which is originally based on analyzing only individual images. The main motivation behind this work was to determine whether groups of similar images are a useful source of information and can improve the automatic annotation and effectiveness of a text-base image retrieval system.

Even a straightforward implementation of either cluster-based approach was demonstrated to have a significantly better annotation and retrieval performance when compared to the original approach. One final comparison, in terms of gain in mean average precision, is reported in Table 7.1.

	CMRM	CQL	CBDM
mAP	0.1354	0.1522(+12.41%)	0.1738(+28.36%)

Table 7.1: Performance improvement measured in Mean Average Precision

In conclusion, our results give consistent evidence that by examining across-collection similarities, which are ignored by an image-by-image anal-

ysis, a cluster-based approach is able to find additional patterns of word-visterm co-occurrences. The immediate benefit from this complementary information is better estimation of the correlations existing between textual and visual components, and consequently, significant improvement of the quality of automatically assigned annotations.

Although we demonstrated the usefulness of clusters in the context of creating annotations for text-based retrieval, image descriptors have a wider range of applications. These include auto-illustration, organizing image collections, facilitating browsing and in general any task which involves users looking at images, since working with images becomes more intuitive for people if visual information is presented together with a short description.

We do not claim that one of the two cluster-based models is superior to the other and should be preferred in any situation. Although Table 7.1 shows the mean average precision of CBDM to be 14.2% higher than that of CQL, the improvement is not statistically significant with p-value 0.316211 according to the Wilcoxon signed rank test. On the contrary, we believe that CQL and CBDM are appropriate in different contexts. CQL could be particularly useful when representations are very short, and therefore small but very compact clusters would likely perform better than individual images. CBDM could be especially effective for very heterogeneous collections, where broad but still coherent clusters would likely perform better than a too outspread overall collection.

A primary characteristic of the cluster-based approach to image annotation, and of CMRM in general, is its independence of the actual document representation. Its conceptual foundation is finding groups of similar objects, which is theoretically possible for all kinds of data. The only requirement so far is that representations are discrete, but this is imposed by the clustering algorithms rather than the overall approach. To implement

cluster-based annotation for continuous objects, we would only need to select a clustering technique capable of processing continuous data. Therefore, the models we described are not limited to image annotation, they can be applied in any situation where we want to automatically interpret objects from one format to another. In the context of IR, such applications include video and audio retrieval where instead of still images we have dense streams of images and sound.

However, our investigation so far has not addressed some of the complexities inherent in using a finite set of examples to learn a general model. Here are some ideas for improvement and future work.

To investigate how specific image properties affect performance, we plan to run a similar set of experiments on a very specialized collection. While the Corel dataset is a general collection with great diversity among its elements, there are many collections which are narrowly specialized and used only in specific settings. An important example are databases of medical images, in which hospitals and research institutes store the medical history of their patients. Typically, a medical collection contains a few generic image types, e.g. X-rays of limbs, torso and head, while diversity within each group is small (because patients are told to stand in front of the radiograph in a particular position). Another significant distinction is that medical images are almost always black-and-white.

In this thesis we did not discuss in details visual features extraction because we used visterm representations that were already generated. But the image processing phase and in particular the selection of visual features should be paid a special consideration since both directly affect how well visterms discriminate between images. Although we did not mention this fact, it turns out that color is the most useful feature in processing the Corel images. With black-and-white medical images, however, color is

meaningless, which prompts the following questions: How do we choose what visual features to use? How do we decide whether a particular feature is useful in describing an image or an image region? Experimenting with a medical collection and implementing all the steps from dividing images into regions to annotating and retrieving images will provide answers or at least an insight to the answers of these questions.

Another significant area for future investigation is clustering. We already compared two conceptually different approaches to creating clusters - hierarchical and non-hierarchical. And our results demonstrated that hierarchical clustering is either better (in the case of CQL) or comparable (in the case of CBDM) to non-hierarchical clustering, while providing us with a fairly sophisticated way to control the quality of clusters. However, additional research is necessary to define the relationship between similarity threshold and cluster coherence more precisely. Advances on this issue could make cluster-based annotation more attractive to people building commercial systems by alleviating the need to set the clustering parameter heuristically or experimentally, which takes up an impractical amount of time and resources. A crucial question in this respect is what characteristics of a collection and its documents affect clustering performance and therefore determine whether cluster-based annotation is successful or not.

Another clustering approach that we want to experiment with is query-specific clustering, where instead of creating the clusters beforehand we take a subset of images that are very similar to the image we want to annotate and cluster only these to obtain specific clusters. This could be particularly useful in large collections where there are many levels of similarity. Consider for example the following situation: We have a huge diverse collection of animal images and an initial static clustering puts images of tigers and lions in the same group. This could be justified if there

are various other, more dissimilar patterns corresponding, for example, to reptiles, birds, etc. However, to annotate an image of a tiger correctly, the system should be able to distinguish between tigers and lions. In this case, query-specific clustering can be especially helpful by assigning the two species to their own group given an image of a tiger.

This last point brings up one more time the fact that similarity is relative and it is usually defined in terms of global rather than local patterns. Two objects can be considered different with respect to their immediate neighborhood and yet they can be considered very similar with respect of the entire space in which they exist. The Cross-media Relevance model, however, does not take into account global similarity patterns, even though it annotates images by finding documents that are similar to it. In this thesis, we showed how this fundamental limitation can be compensated by extracting and incorporating information from groups of similar images that have been created by examining the structure of the entire collection. Furthermore, this information improves the quality of automatic annotations and image retrieval performance.

Bibliography

[1] K. Barnard, P. Duygulu, N. de Freitas, D. Forsyth, D. Blei, and M. I. Jordan. Matching Words with Pictures. *Journal of Machine Learning Research*, 3:1107–1135, 2003.

[2] K. Barnard and D. Forsyth. Learning the Semantics of Words and Pictures. In *Proceedings of the 8th IEEE International Conference on Computer Vision*, pages 408–415, 2001.

[3] A. Berger, S. A. Della Pietra, and V. J. Della Pietra. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1), 1996.

[4] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, 1993.

[5] P. Carbonetto and N. de Freitas. Why Can't José Read? - The Problem of Learning Semantic Associations in a Robot Environment. In *NAACL Human Language Technology Conference Workshop on Learning Word Meaning from Non-Linguistic Data*, 2003.

[6] Y. Choi and E. M. Rasmussen. Users' Relevance Criteria in Image Retrieval in American History. *Information Processing & Management*, 38:695–726, 2001.

[7] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41:391–407, 1990.

[8] P. Duygulu, K. Barnard, N. de Freitas, and D. Forsyth. Object Recognition as Machine Translation: Leaning a Lexicon for a Fixed Image Vocabulary. In *Proceedings of 7th European Conference on Computer Vision*, pages 97–112, 2002.

[9] M. H. Hearst and J. O. Pedersen. Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results. In *Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 76–84, 1996.

[10] A. Hughes, T. Willkens, B. M. Wildemuth, and G. Marchionini. An Eyetracking Study of How People View Digital Video Surrogates. In *Proceedings of the 2nd International Conference on Image and Video*, pages 259–268, 2003.

[11] G. Iyengar, P. Duygulu, S. Khudanpur, D. Klakow, R. Manmatha, H. J. Nock, S. Feng, P. Ircing, B. Pytlik, P. Virga, M. Krause, and D. Petkova. Joint Visual-Text Modeling for Multimedia Retrieval. Technical report, Center for Language and Speech Processing, Johns Hopkins University, 2004.

[12] F. Jelinek and R. L. Mercer. *Interpolated Estimation of Markov Source Parameters from Sparse Data*. 1980.

[13] J. Jeon, V. Lavrenko, and R. Manmatha. Automatic Image Annotation and Retrieval using Cross-Media Relevance Models. In *Proceedings of the 26th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 119–126, 2003.

[14] J. Jeon and R. Manmatha. Using Maximum Entropy for Automatic Image Annotation. In *Proceedings of the 3rd International Conference on Image and Video Retrieval*, pages 24–32, 2004.

[15] V. Lavrenko, R. Manmatha, and J. Jeon. A Model for Learning the Semantics of Pictures. In *Proceedings of the 17th Annual Conference on Neural Information Processing Systems*, pages 553–560, 2003.

[16] X. Liu and W. B. Croft. Cluster-Based Retrieval using Language Models. In *Proceedings of the 27th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 186–193, 2004.

[17] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[18] Y. Mori, H. Takanashi, and R. Oka. Image-to-word Transformation based on Dividing and Vector Quantizing Images with Words. In *MISRM 1st International Workshop on Multimedia Intelligent Storage and Retrieval Management*, 1999.

[19] K. Rodden, W. Basalaj, D. Sinclair, and K. R. Wood. Does Organisation by Similarity Assist Image Browsing? In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 2001.

[20] University of California at Berkeley. Corel database website at `http://elib.cs.berkeley.edu/photos/corel/`.

[21] C. J. van Rijsbergen. *Information Retrieval*. London: Butterworths, 1979.

[22] G. Wei and I. K. Sethi. Omni-face Detection for Video/Image Content Description. In *Proceedings of the ACM Workshop on Multimedia Information Retrieval*, 2000.

[23] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers, 1999.

[24] C. Zhai and J. Lafferty. A Study of Smoothing Mehtods for Language Models Applied to Ad Hoc Information Retrieval. In *Proceedings of the 24th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 334–342, 2001.