A Semantics-Based Approach for Speech Annotation of Images

Dmitri V. Kalashnikov, Sharad Mehrotra, *Member*, *IEEE*,
Jie Xu, and Nalini Venkatasubramanian, *Member*, *IEEE*

Abstract—Associating textual annotations/tags with multimedia content is among the most effective approaches to organize and to support search over digital images and multimedia databases. Despite advances in multimedia analysis, effective tagging remains largely a manual process wherein users add descriptive tags by hand, usually when uploading or browsing the collection, much after the pictures have been taken. This approach, however, is not convenient in all situations or for many applications, e.g., when users would like to publish and share pictures with others in real time. An alternate approach is to instead utilize a speech interface using which users may specify image tags that can be transcribed into textual annotations by employing automated speech recognizers. Such a speech-based approach has all the benefits of human tagging without the cumbersomeness and impracticality typically associated with human tagging in real time. The key challenge in such an approach is the potential low recognition quality of the state-of-the-art recognizers, especially, in noisy environments. In this paper, we explore how semantic knowledge in the form of co-occurrence between image tags can be exploited to boost the quality of speech recognition. We postulate the problem of speech annotation as that of disambiguating among multiple alternatives offered by the recognizer. An empirical evaluation has been conducted over both real speech recognizer's output as well as synthetic data sets. The results demonstrate significant advantages of the proposed approach compared to the recognizer's output under varying conditions.

Index Terms—Using speech for tagging and annotation, using semantics to improve ASR, maximum entropy approach, correlation-based approach, branch and bound algorithm.

1 Introduction

Increasing popularity of digital cameras and other multimedia capture devices has resulted in the explosion of the amount of digital multimedia content. Annotating such content with informative tags is important to support effective browsing and search. Several methods could be used for such annotation, as explained below.

For image repositories, the first way to annotate pictures is to build a system that relies entirely on visual properties of images. The state-of-the-art image annotation systems of that kind work well in detecting generic object classes: car, horse, motorcycle, airplane, etc. However, there are limitations associated with considering only image content for annotation. Specifically, certain classes of annotations are more difficult to capture. These include location (Paris, California, San Francisco, etc.), event (birthday, wedding, graduation ceremony, etc.), people (John, Jane, brother, etc.), and abstract qualities referring to objects in the image (beautiful, funny, sweet, etc.).

The second and more conventional method of tagging pictures is to rely completely on human input. This approach has several limitations too. For instance, many cameras do not have an interface to enter keywords. Even if they do, such a tagging process might be cumbersome and inconvenient to do right after pictures are taken. Alternatively, a user could tag images at a later time while either uploading them to a repository or browsing the images. Delay in tagging may result in a loss of context in which the picture was taken (e.g., user may not remember the names of the people/structures in the image). Furthermore, some applications dictate that tags be associated with images right away.

The third possibility for annotating images uses speech as a modality to annotate images and/or other multimedia content. Most cameras have built-in microphone and provide mechanisms to associate images with speech input. In principle, some of the challenges associated with both fully automatic annotation as well as manual tagging can be alleviated if the user uses speech as a medium of annotation. In an ideal setting, the user would take a picture and speak the desired tags into the device's microphone. A speech recognizer would transcribe the audio signal into text. The speech to text transcription could either happen on the device itself or be done on a remote machine. This text can be used in assigning tags to the image. The proposed solution is useful in general scenarios, where users might want to use a convenient speech interface for assigning descriptive textual tags to their images. Such systems also can play a critical role in applications that require real-time triaging of images to a remote site for further analysis, such as reconnaissance and crisis response applications.

All three aforementioned tagging approaches are not competing and, in practice, can complement each other. For instance, tags added via speech can be enhanced at a later

• *The authors are with the Department of Computer Science, Bren School of Information and Computer Science, University of California at Irvine, Irvine, CA 92617. E-mail: {dvk, sharad, jiex, nalini}@ics.uci.edu.*

Manuscript received 4 July 2009; revised 23 Feb. 2010; accepted 14 Apr. 2010; published online 23 Sept. 2010.
Recommended for acceptance by V.S. Subrahmanian.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2009-07-0528.
Digital Object Identifier no. 10.1109/TKDE.2010.185.

point by adding more tags manually. In this paper, however, we will primarily focus on exploring the advantages of the third, Speech Tagging Interface (STI), technology.

1.1 Motivating Application Domain

While STI technology is of value in a variety of application domains, our work is motivated by the emergency response domain. In particular, we have explored STI in the context of the Situational Awareness for Firefighters (SAFIRE) project wherein our goal is to enhance the safety of the public and firefighters from fire and related hazards [28]. We are developing situational awareness technologies that provide firefighters with synchronized real-time information. These tools are expected to enhance safety and facilitate decision making for firefighters and other first responders. The ultimate goal is to develop an information-and-control-panel prototype called the Fire Incident Command Board. This device will combine new and existing hardware and software components that can be customized to meet the needs of field incident commanders. FICB tools will allocate resources, monitor status and locale of personnel, and record and interpret site information. The FICBs will integrate and synchronize sensors and other information flows from the site and provide customized views to individual users while seamlessly interacting with each other.

One of the functionalities of the system we are developing is image-triaging capability. Consider, for instance, a mission critical setting in which a picture of a disaster site is taken by the first responder (e.g., a firefighter). It could help to create real-time situational awareness if triaged to the appropriate personnel that deal with response operations. Building such a real-time triaging capability requires the descriptive tags to be associated with the image (e.g., details about the location, victims, exit routes, etc.) in real time.

One of the biggest challenges facing such speech annotation systems is the accuracy of the underlying speech recognizer. Even speaker-dependent recognition systems can make mistakes in noisy environments. If the recognizer's output is considered for annotation "as is," then poor speech recognition will lead to poor quality tags which, in turn, lead to both false positives as well as false negatives in the context of triaging.

1.2 Our Approach

Our work addresses the poor quality of annotations by incorporating outside semantic knowledge to improve interpretation of the recognizer's output, as opposed to blindly believing what the recognizer suggests. Most speech recognizers provide alternate hypotheses for each speech utterance of a word, known as the N-best list for the utterance. We exploit this fact to improve interpretation of speech output. Our goal is to use semantic knowledge in traversing the search space that results from these multiple alternatives in a more informative way, such that the right annotation is chosen from the N-best list for each given utterance. We show that by doing so, we can improve the quality of speech recognition and thereby improve the quality of the image tag assignment.

The semantic knowledge can potentially come from a variety of sources. Different sources can be useful in interpreting different types of utterances. For instance, knowledge of geographic location of the picture could help in interpreting speech utterances involving names of streets, cities, and states. Domain lexicons can help improve interpretation of utterances specific to a domain, such as utterances concerning Air Traffic Control Systems. A user's address book can help improve recognition of names of people, and so on. While in principle, multiple sources of semantics can be considered at the same time, we propose a framework that considers semantics acquired from a previously annotated corpus of images. We show that understanding semantic relationships between tags in an existing corpus can improve interpretation of speech recognizer output concerning a new image. We show that the speech interpretation problem requires addressing several issues. These include designing a mathematical formulation for computing a "score" for a sequence of words and developing efficient algorithms for disambiguation among word alternatives. Specifically, the main contributions of this paper are:

- **ME score**. A probabilistic model for computing the probability of a given combination of tags that builds on Lidstone's Estimation and Maximum Entropy approaches (Section 4).
- **CM score**. A correlation-based way to compute a score of a tag sequence to assess the likelihood of a particular combination of tags (Section 5).
- **Branch and bound algorithm.** A branch and bound algorithm for efficient search of the most likely combinations of tags (Section 6).
- **Empirical evaluation.** An extensive empirical evaluation of the proposed solution (Section 8).

The rest of the paper is organized as follows: We start by presenting the related work in Section 2. We then formally define the problem of annotating images using speech in Section 3. Next, we explain how semantics can be used to score a sequence of tags, first using a Maximum Entropy approach in Section 4 and then using correlation-based approach in Section 5. Section 6 then describes how these computations can be speed up using a branch and bound algorithm. The extensions of the proposed framework are discussed in Section 7. The proposed approach is extensively tested in Section 8. Finally, we conclude in Section 9.

2 Related Work

In this section, we start by discussing work related to other speech-based annotation systems in Section 2.1. We then cover some of closely related solutions that do not deal directly with speech in Section 2.2. Finally, in Section 2.3, we highlight the contribution of this paper compared to its preliminary version.

2.1 Speech Annotation Systems

Several speech annotation systems have been proposed that utilize speech for annotation and retrieval of different kinds of media [3], [19], [20], [21], [31], [32]. In [20], [21], the authors propose to investigate a simple and natural extension of the way people record video. It allows people to speak out annotations during recording. Spoken annotations are then transcribed to text by a speech recognizer. This approach,

however, requires certain syntax of annotations, and specifically that each content-descriptive free speech annotation is preceded by a keyword specifying the kind of annotation and its associated temporal validity. Our approach does not require any particular syntax of annotations. The system does not utilize any outside knowledge to improve recognition accuracy.

The authors in [31] propose a multimedia system for semiautomated image annotation. It combines advances in speech recognition, natural language processing, and image understanding. It uses speech to describe objects or regions in images. However, to resolve the limitation of speech recognizer, it requires several additional constraints and tools:

- Constraining the vocabulary and syntax of the utterances to ensure robust speech recognition. The active vocabulary is limited to 2,000 words.
- Avoiding starting utterances with such words as "this" or "the." These words might promote ambiguities.
- Providing an editing tool to permit correction of speech transcription errors.

The approach in [20], [21], [31] all utilize speech for annotation. However, instead of utilizing outside semantics, they all require certain constraints on the structure of annotation to help speech recognition. Our framework place little limitation on the syntax of annotation, but make efforts to incorporate outside semantic information.

The approach in [3] employs structured speech syntax for tagging. It segments each speech annotation into four parts: *Event*, *Location*, *People*, and *Taken_On*. During annotation, it retains the entire N-best list associated with every utterance. The approach utilizes two different query expansion techniques for retrieval:

1. Using an outside source of knowledge, such as a thesaurus or a concept hierarchy, it maps each query term to multiple possibilities.
2. It utilizes the N-best lists to mitigate the effects of the ASR's substitution errors. By observing the type of mistakes the recognizer makes, the approach uses the items in the N-best list to compute the probability of each item in the list conditioned on the actual utterance. This conditional probability plays a role in computing the image-query similarity.

Of the systems listed above, only [3] works in an N-best list framework. We clarify some key differences between [3] and our work:

1. The task of the approach in [3] is to employ semantics in improving precision and recall from a retrieval standpoint, using the two query expansion techniques listed above. The goal of our approach is to improve the quality of recognition itself in the context of speech annotation, which would naturally translate into improved quality of tags.
2. The approach in [3] addresses the annotation problem on a personal photo collection. The author focus on the case where each annotation can be segmented into (*Event*, *Location*, *People*, and *Taken_On*) classes, which is consistent with the types of tags that people provide on the data set they study. We, on the other hand, look at the problem of annotation of photos where the spoken tags are either nouns or adjectives in the English language. Given the nature of the tags we consider, we do not impose the kind of structure that is used in [3], which might be too restrictive in general settings. For instance, many annotations of images in various photo sharing applications such as Flickr (for example, annotations *butterfly*, *garden*, *rose*, *beautiful*, and *nature*), do not readily lend themselves to being divided into *Event*, *Location*, *People*, and *Taken_On*.

2.2 Nonspeech Image Annotation

Due to practical significance of the problem, many different types of image tagging techniques have been developed. In the previous section, we have already reviewed techniques that utilize speech for annotation. In this section, we will overview those that do not employ speech for that purpose. Observe that while the goal of our problem is to derive image tags from the corresponding speech utterances, the goal of the techniques discussed in this section is naturally different, since they do not use speech. Typically, their goal is to derive tags automatically from image features or to assign them manually by the user. Because of the difference of the goals, the techniques mentioned in this section are not competing to our approach. They are rather complementary as they can be leveraged further to better interpret utterances of spoken keywords, but developing techniques that can achieve this is beyond the scope of this paper.

Many content-based annotation algorithms have been proposed to annotate images automatically, based on the content of images and without using speech. Such algorithms usually generate several candidate annotations by learning the correlation between keywords and visual content of images. Given a set of images annotated with a set of keywords that describe the image content, a statistical model is trained to determine the correlation between keywords and visual content. This correlation can be used to annotate images that do not have annotations. Candidate annotations are then refined using semantics. For instance, [10], [33], [34] utilize certain semantic information to filter out irrelevant candidates, e.g., by using WordNet. The approaches in [10], [33], [34] are based on the basic assumption that highly correlated annotations should be preserved and noncorrelated annotations should be removed. In addition, certain combination of words can be preferred, or filtered out, by using N-gram or other language model and NLP-based techniques, especially, in the scenarios where not just keywords/tags, but complete sentences are used for annotations [35].

The correlation between keywords and image features can be also captured by learning a statistical model, including Latent Dirichlet Allocation (LDA) [2] and Probabilistic Latent Semantic Analysis (PLSA) [9]. Annotations for unlabeled images are generated by employing these models [24]. In [24], the authors encode image features as visual keywords. Images are modeled by concatenated visual keywords and if any, annotations. Semantic Analysis (LSA) is applied to compute semantic similarity between an

unannotated image and the annotated image corpus. The annotation is then propagated from the ranked documents. In addition, PLSA is applied to compute distribution of the terms of the vocabulary given an unannotated image.

Social tagging is a manual image tagging approach where a community of users participate in tagging of images [18]. Different users can tag the same image and the end tags for an image are decided according to some policy. For instance, when a certain number of users submit the same tag for an image, the tag is assigned to the image.

Diaz et al. in [8] investigates ways to improve tag interoperability across various existing tagging systems by providing a global view of the tagging sites. By utilizing a query language, it is possible to assign new tags, change existing ones, and perform other operations. The system uses RDF graph as its data model and assumes that existing tagging systems will eventually become RDF graph providers.

2.3 Our Past Work

The differences of this paper compared with its initial version [7] include:

1. Related work is now covered (Section 2);
2. More in-depth coverage of the problem definition, including the pseudocode of the mentioned algorithm (Section 3);
3. More in-depth coverage of the Max Entropy solution (Section 4);
4. More in-depth coverage of correlation, including new material related to indirect correlation and correlation and membership scores (Section 5);
5. The Branch and Bound algorithm that makes the approach scale to large data sets, thus making it feasible in practice (Section 6);
6. A method for combining the results of the global and local models, that leads to higher quality of annotations (Section 7);
7. Five new experiments that study various aspects of the proposed solution (Section 8). Some of our past entity resolution work is also related, but not directly applicable and uses different methodologies [4], [5], [6], [12], [13], [14], [15], [16], [17], [25], [26].

3 Notation and Problem Definition

We consider a setting wherein the user intends to annotate an image with a sequence $G = (g_1, g_2, \ldots, g_K)$ of K ground truth tags. Each tag g_i can be either a single word or a short phrase of multiple words, such as Niagara Falls, Golden Gate Bridge, and so on. Since a tag is typically a single word, we will use "tag" and "word" interchangeably. Table 1 summarizes the notation.

3.1 N-Best Lists

To accomplish the annotation task, the user provides a speech utterance for each tag g_i for $i = 1, 2, \ldots, K$, which are then processed by a speech recognizer. Similar to the segmentation assumptions that Chen et al. make in [3], we assume that the recognizer is trained to recognize a delimiter between each of these K utterances and thus it will know the correct number of utterances K. The recognizer's task is to recognize these words correctly so that the correct tags are assigned to the image. However, speech recognition is prone to errors, especially, in noisy environments and for unrestricted vocabularies and the recognizer might propose several *alternatives* for one utterance of a word. Consequently, the output of the recognizer is a sequence $\mathcal{L} = (L_1, L_2, \ldots, L_K)$ of K N-best lists for the utterances.

TABLE 1
Notation

Notation	Meaning
$G = (g_1, g_2, \ldots, g_K)$	Sequence of K ground truth tags
g_i	i-th ground truth tag
$\mathcal{L} = (L_1, L_2, \ldots, L_K)$	Set of K N-best lists
$L_i = (w_{i1}, w_{i2}, \ldots, w_{iN})$	i-th N-best list
w_{ij}	j-th word in i-th N-best list
$W = (w_1, w_2, \ldots, w_K)$	Sequence of tags: $w_1, w_2, \ldots, w_K$ are tags associated with an image
$n(w_1, w_2, \ldots, w_n)$	Number of images whose annotations include tags $w_1, w_2, \ldots, w_n$
N_I	Overall number of images
$c(w_i, w_j)$	Direct corr. between tags w_i & w_j
$A(w_i, w_j)$	Indir. correl. between w_i and w_j
$\mathcal{G}$	Direct correlation graph
$\mathcal{G}_{ind}$	Indirect correlation graph

Each N-best list $L_i = (w_{i1}, w_{i2}, \ldots, w_{iN})$ consists of N words that correspond to the recognizer's alternatives for word g_i. Observe that list L_i might not contain the ground truth word g_i. The words in an N-best lists L_i are typically output in a ranked order. Thus, when the recognizer has to commit to a concrete single word for each utterance, it would set $N = 1$ and output $(w_{11}, w_{21}, \ldots, w_{K1})$ as its answer. While w_{i1} has the highest chance of being the correct word, in practice, it is often not the case, leading to a possibility of improving the quality of annotation.

3.2 Sequences

Let us define a *sequence* as a K-dimensional vector $W = (w_1, w_2, \ldots, w_K)$, where w_i can be of three types:

1. $w_i \in L_i$, that is, w_i is one of the N words from list L_i;
2. $w_i = \texttt{null}$, which encodes the fact that the algorithm believes list L_i does not contain g_i;
3. $w_i =$ "$-$", that is, the algorithm has not yet decided the value of the ith tag.

The *cardinality* $|W|$ of sequence W is defined as the number of the elements of the first category that the sequence has: $|W| = |\{w_i \in W : w_i \in L_i\}|$. Sequence W is an *answer* sequence or a *complete* sequence, if none of its elements w_i is equal to "$-$". In other words, an answer sequence cannot contain undecided tags, only words from the N-best lists or `null` values.

3.3 Answer Quality

Now we can define the *quality* of sequence $W = (w_1, w_2, \ldots, w_K)$ by adapting the standard IR metrics of precision, recall, and F-measure [1]. Namely, if $|W| = 0$ then $Precision(W) = Recall(W) = 0$. If $|W| > 0$ then

$$Precision(W) = \frac{|W \cap G|}{|W|}$$

```
COMPUTE-ANSWER(L)
1  W* ← ∅   // Best sequence
2  s* ← 0   // Best score
3  for each w_{1j1} ∈ L_1, w_{2j2} ∈ L_2, ..., w_{KjK} ∈ L_K do
4     W ← (w_{1j1}, w_{2j2}, ..., w_{KjK})
5     s ← GET-SCORE(W)
6     if s > s* do
7        W* ← W
8        s* ← s
9  W* ← PICK-NULLS(W*)
10 return W*
```

Fig. 1. Overall algorithm with naïve enumeration of sequences.

and $Recall(W) = \frac{|W \cap G|}{|G|} = \frac{|W \cap G|}{K}$, where $|W \cap G|$ is the number of w_i such that $w_i = g_i$. The F-measure is computed as the harmonic mean of the precision and recall. Thus, our **goal** can be viewed as that of designing an algorithm that produces high quality answer for any given $\mathcal{L}$.

Having defined the quality of answer, we can make several observations. First, for a given $\mathcal{L}$, the best answer sequence is the sequence $W = (w_1, w_2, \ldots, w_K)$ such that $w_i = g_i$ if $g_i \in L_i$ and $w_i = \texttt{null}$ if $g_i \notin L_i$. Second, there is a theoretic upper bound on the achievable quality of any sequence W for a given $\mathcal{L}$. Specifically, assume that only M out of K N-best lists contain the ground truth tags, where $M \leq K$. Then, the maximum reachable value of $|W \cap G|$ is M. Thus, if $M = 0$ then for any answer W it follows that $Precision(W) = Recall(W) = 0$. If $M > 0$, then the maximum reachable precision is $\frac{M}{M} = 1$ and maximum recall is $\frac{M}{K}$ that is less than 1 when $M < K$.

3.4 Overall Goal

Now we have developed the necessary notation and we can formally define the problem. The overall goal is to design an algorithm that when given an N-best list set $\mathcal{L}$ produces an answer sequence that is as close to the ground truth G as possible. The effectiveness of different algorithms will be compared using the average F-measure quality metric. Here, given N-best list sets $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_k$ and the corresponding answers computed by the algorithm $W_1^*, W_2^*, \ldots, W_k^*$, the average F-measure is defined as $\frac{1}{k}\sum_{i=1}^{k} F(W_i^*)$.

Since the ground truth is unknown, the goal becomes that of finding an algorithm that reaches high quality of answers. We will specifically focus on a large class of algorithms that consider only answer sequences, as defined in Section 3.2, as possible answers. We will pursue a two-step strategy to solve the problem:

Step 1: Maximizing score. Let $\mathcal{W}_{\mathcal{L}} = \{W\}$ be the set of all N^K possible answer sequences given $\mathcal{L}$. For each such sequence $W \in \mathcal{W}_{\mathcal{L}}$, the algorithm uses a scoring strategy to compute its score $S(W)$. First, the algorithm finds sequence W^* by solving an optimization problem $W^* = \text{argmax}_{W \in \mathcal{W}_{\mathcal{L}}} S(W)$ where the task is to locate sequence $W \in \mathcal{W}_{\mathcal{L}}$ that maximizes the score $S(W)$. In Sections 4 and 5, we will present two such scoring strategies that get high-quality answers by employing the maximum entropy and correlation-based techniques.

Step 2: Detecting nulls. The algorithm then applies a null detection procedure to W^* to compute its final answer, as will be elaborated in Section 7.

TABLE 2
Sample N-Best Lists $\mathcal{L} = (L_1, L_2, L_3, L_4, L_5)$

L_1	L_2	L_3	L_4	L_5
w_{11} =pain	w_{21} =prose	w_{31} =**garden**	w_{41} =**flower**	w_{51} =sad
w_{12} =**Jane**	w_{22} =nose	w_{32} =harden	w_{42} =power	w_{52} =wad
w_{13} =lane	w_{23} =**rose**	w_{33} =jordan	w_{43} =shower	w_{53} =bad
w_{14} =game	w_{24} =crows	w_{34} =pardon	w_{44} =tower	w_{54} =dad

Fig. 1 outlines a naive implementation of the approach. For the class of algorithms we consider, the overall goal translates into that of designing a scoring and null detecting strategies that reach high answer quality.

3.5 Notational Example

As an example, suppose that the user takes a picture of her friends *Jane* in a garden full of roses, and provides the utterances of $K = 5$ words: $G = (g_1 = Jane,\ g_2 = rose,\ g_3 = garden,\ g_4 = flower,\ g_5 = red)$. Then, the corresponding set of five N-best lists for $N = 4$ could be as illustrated in Table 2. If the recognizer has to commit to a single word per utterance, its output would be $(pain, prose, garden, flower, sad)$. That is, only "garden" and "flower" would be chosen correctly. This motivates the need for an approach that can disambiguate between the different alternatives in the list. For the types of the algorithms being considered, the best possible answer would be $(Jane, rose, garden, flower, \texttt{null})$. The last word is null since list L_5 does not contain the ground truth tag $g_5 = red$. Therefore, the maximum achievable precision is 1 and recall is $\frac{4}{5}$. Suppose some approach is applied to this case, and its answer is $W = (Jane, rose, garden, power, \texttt{null})$, that is, it picks "power" instead of "flower" and thus only "Jane," "rose," and "garden" tags are correct. Then, $Precision(W) = \frac{3}{4}$ and $Recall(W) = \frac{3}{5}$.

4 Using Maximum Entropy to Score a Sequence of Words

Section 3 has explained that given a sequence of N-best lists $\mathcal{L}$ the algorithm chooses its answer sequence W^* as the one that maximizes the score $S(W)$ among all possible answer sequences $W \in \mathcal{W}_{\mathcal{L}}$. In this section, we discuss a principled way to assign a score to a given sequence W.

The ME approach covered in this section computes the score $S_{ME}(W)$ of sequence $W = (w_1, w_2, \ldots, w_K)$ as the joint probability $S_{ME}(W) = P(w_1, w_2, \ldots, w_K)$ for an image to be annotated with tags $w_1, w_2, \ldots, w_K$. This probability is inferred based on how images have been annotated in past data.

4.1 Maximum Likelihood Estimation

The main challenge is to compute this joint probability. Ideally, whenever possible, we would want to estimate the known joint probabilities directly from data. For instance, we could consider the Maximum Likelihood Estimation (MLE) approach for such an estimation:

$$P(w_1, w_2, \ldots, w_K) = \frac{n(w_1, w_2, \ldots, w_K)}{N_I}. \quad (1)$$

In this formula, $n(w_1, w_2, \ldots, w_K)$ is the number of images annotated with tags $w_1, w_2, \ldots, w_K$ and N_I is the overall

number of images. However, the MLE is known to be impractical in problem setting like ours since it would require extremely large training data set. To illustrate the problem, consider a simple scenario where each image is annotated with exactly two tags, such that each tag is taken from a small English dictionary of 10^4 words. Thus, there are $C^2_{10^4}$ distinct annotations possible, which is in the order of $5 \cdot 10^7$. Therefore, to reliably estimate the probabilities based on counts for even this simple scenario of two-tag annotations only, we will need a corpus of more than $5 \cdot 10^7$ images, which makes the approach impractical. In turn, for a realistic training data sample, $n(w_1, w_2, \ldots, w_K)$ would be frequently equal to zero leading to incorrect assignments of probabilities. This is especially the case for larger values of K, e.g., $K \geq 3$.

4.2 Lidstone's Estimation

To overcome the above problem in estimating $P(w_1, w_2, \ldots, w_K)$, we employ a combination of the Lidstone's Estimation (LE) and Maximum Entropy (ME) approaches [11], [22], [23]. The LE method addresses some of the limitations of MLE by making an assumption of uniform priors on unobserved sequences:

$$P(w_1, w_2, \ldots, w_K) = \frac{n(w_1, w_2, \ldots, w_K) + \lambda}{N_I + \lambda |V|^K}. \quad (2)$$

Here, $|V|$ is the number of possible words in the vocabulary and λ is a small value that is added to each count. The most common ways of setting λ are 1) $\lambda = 1$, known as the Laplace estimation, 2) $\lambda = 0.5$, known as the Expected Likelihood Estimation (ELE), or 3) learning λ from data.

The limitation of the LE approach is that for larger values of K, it is likely that $n(w_1, w_2, \ldots, w_K) = 0$. Thus, the LE will assign the same probability of $\frac{\lambda}{N_I + \lambda |V|^K}$ to most of $P(w_1, w_2, \ldots, w_K)$ whereas a better estimate can be computed, for instance, by using the Maximum Entropy approach.

4.3 Maximum Entropy Approach

The ME approach reduces the problem of computing $P(w_1, w_2, \ldots, w_K)$ to a *constrained optimization problem*. It allows to compute joint probability $P(w_1, w_2, \ldots, w_K)$ based on only the values of known correlations in data. The approach hinges on the information-theoretic notion of entropy [29]. For a probability distribution $P = (p_1, p_2, \ldots, p_n)$, where $\sum p_i = 1$, the entropy $H(P)$ is computed as $H(P) = -\sum_{i=1}^{n} p_i \log p_i$ and measures the uncertainty associated with P. Entropy $H(P)$ reaches its minimal value of zero in the most certain case where $p_i = 1$ for some i and $p_j = 0$ for all $j \neq i$. It reaches its maximal value in the most uncertain uniform case where $p_i = \frac{1}{n}$ for $i = 1, 2, \ldots, n$.

Let us first introduce some useful notation necessary to explain the ME approach. We will use a support-based method to decide whether the probability can be estimated directly from data [11], [22]. Specifically, if $K = 1$, or if $K \geq 2$ and $n(w_1, w_2, \ldots, w_K) \geq k$, where k is a positive integer value, then there is sufficient support to estimate the joint probability directly from data and $P(w_1, w_2, \ldots, w_K)$ is computed using (2). We will refer to such $P(w_1, w_2, \ldots, w_K)$ as *known* probabilities. Cases of $P(w_1, w_2, \ldots, w_K)$ where $K \geq 2$ but $n(w_1, w_2, \ldots, w_K) < k$ do not have sufficient support. They will be handled by the ME approach instead of (2). We will refer to them as *unknown* probabilities.

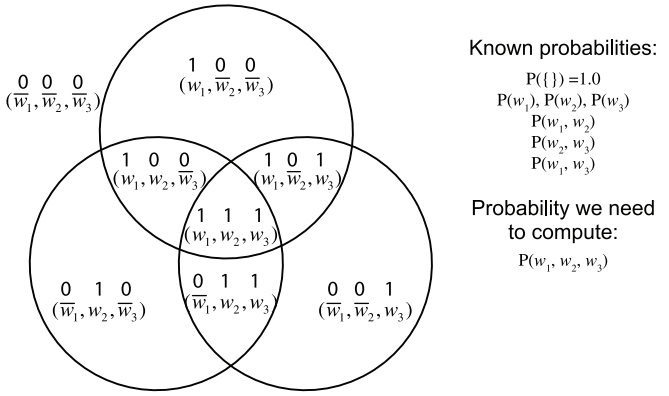


Fig. 2. Probability space.

To compute $P(w_1, w_2, \ldots, w_K)$, the ME approach considers the power set $\mathcal{S}$ of set $\{w_1, w_2, \ldots, w_K\}$, that is, the set of all its subsets. For instance, the power set of $\{w_1, w_2, w_3\}$ is $\{\{\}, \{w_1\}, \{w_2\}, \{w_1, w_2\}, \{w_2, w_3\}, \{w_1, w_2, w_3\}\}$. We can observe that for some of the subsets $S \in \mathcal{S}$ the probability $P(S)$ will be known and for some it will be unknown. Let $\mathcal{T}$ be the *truth* set, i.e., the set of subsets for which $P(S)$ is known: $\mathcal{T} = \{S \in \mathcal{S} : P(S) \text{ is known}\}$. The values of $P(S)$, where $S \in \mathcal{T}$, will be used to define the constraints for the constrained optimization problem.

To compute $P(w_1, w_2, \ldots, w_K)$, the algorithm considers *atomic annotation descriptions*, which are tuples of length K, where the ith element can be only either w_i or $\overline{w}_i$. Here, w_i means tag w_i is present in annotations and $\overline{w}_i$ means w_i is absent from them. For instance, description $(w_1, w_2, \overline{w}_3)$ refers to all image annotations where tags w_1 and w_2 are present and w_3 is absent. Each such description can be encoded with a help of a bit string b, where 1 corresponds to w_i and 0 to $\overline{w}_i$. For instance, $(w_1, w_2, \overline{w}_3)$ can be encoded as $b = 110$. Let A_S be the atom set for S, defined as the set of all possible bit strings of size K such that for each $b \in A_S$ it holds that if $w_i \in S$ then $b[i] = 1$, for $i = 1, 2, \ldots, K$. For instance, for $K = 3$ and $S = \{w_1, w_2\}$ set $A_S = \{110, 111\}$, whereas for $K = 3$ and $S = \{w_2\}$ set $A_S = \{010, 011, 110, 111\}$.

Let x_b denote the probability to observe an image annotated with the tags that correspond to bit string b of length K. Fig. 2 illustrates the probability space with respect to all x_b for the case where $K = 3$. Then, in the context of ME approach, our goal of determining $P(w_1, w_2, \ldots, w_K)$ reduces to solving the following constrained optimization problem:

$$\begin{cases} \text{Maximize } Z = -\sum_{b \in B} x_b \log x_b \\ \text{subject to} \\ \quad \sum_{b \in A_S} x_b = P(S), \text{ for all } S \in \mathcal{T}, \\ \text{and} \\ \quad x_b \geq 0, \text{ for all } b. \end{cases} \quad (3)$$

Solving it will give us the desired $P(w_1, w_2, \ldots, w_K)$ which corresponds to $x_{11\ldots1}$. The constrained optimization problem can be solved efficiently by the method of Lagrange multipliers to obtain a system of optimality equations. Since the entropy function is concave, the

$$
\begin{aligned}
x_{000} + x_{001} + x_{010} + \cdots + x_{111} &= 1.0\\
x_{100} + x_{101} + x_{110} + x_{111} &= 0.2\\
x_{010} + x_{011} + x_{110} + x_{111} &= 0.3\\
x_{001} + x_{011} + x_{101} + x_{111} &= 0.3\\
x_{110} + x_{111} &= 0.12\\
x_{101} + x_{111} &= 0.13\\
x_{011} + x_{111} &= 0.23
\end{aligned}
$$

and

$$x_{000} \geq 0, x_{001} \geq 0, x_{010} \geq 0, \ldots, x_{111} \geq 0$$

Fig. 3. Constraints for the ME example.

optimization problem has a unique solution [27]. We employ the variant of the iterative scaling algorithm used by [23] to solve the resulting system.

The advantage of using ME approach is that it takes into account all the existing information, that is, all known marginal and joint probabilities. It also tries to avoid bias in computing $P(w_1, w_2, \ldots, w_K)$ by making uniformity assumptions when information on particular correlations is absent while at the same time trying to satisfy all the constraints posed by the known correlations.

Example. Suppose that we need to compute $P(w_1, w_2, w_3)$. Assume that using the support method we determine that the known probabilities are: the trivial case $P(\{\}) = 1.0$; the marginals $P(w_1) = 0.2, P(w_2) = 0.3$, $P(w_3) = 0.3$; and the pairwise joints $P(w_1, w_2) = 0.12$, $P(w_1, w_3) = 0.13, P(w_2, w_3) = 0.23$. Then, the constraints of the corresponding system are illustrated in Fig. 3. After solving this system, we will get $P(w_1, w_2, w_3) = x_{111} = 0.11$.

5 Using Correlation to Score a Sequence of Words

In this section, we define the notion of the correlation $c(w_i, w_j)$ between any pair of words w_i and w_j. We will use this notion for a variety of purposes. First, it will allow us to define the notion of the correlation score $C(W)$ of a sequence of words W (Section 5.3). Unlike ME score, computing which is exponential in the number of words in the sequence, the correlation score can be computed efficiently. Second, the correlation score is amenable to quick upper and lower bounding, which will enable to speed up the overall algorithm illustrated in Fig. 1. This is achieved by designing a Branch and Bound algorithm that avoids enumerating all possible N^K sequences leading to very significant speedup of the overall algorithm (Section 6). Finally, we will use correlation to create a method for detecting `null` cases (Section 7.1).

5.1 Direct Correlation

Let w_i and w_j be the ith and jth words from a vocabulary V. Then, correlation $c(w_i, w_j)$ is defined as the Jaccard similarity:

$$c(w_i, w_j) = \begin{cases} \dfrac{n(w_i, w_j)}{n(w_i) + n(w_j) - n(w_i, w_j)}, & \text{if } n(w_i, w_j) > 0;\\ 0, & \text{if } n(w_i, w_j) = 0. \end{cases} \tag{4}$$

In this formula, $n(w_i, w_j)$ is the number of images whose annotation include tags w_i and w_j and $n(w_i)$ is the number of images that have tag w_i. The value $c(w_i, w_j)$ is always in $[0, 1]$ interval. It measures how similar the set of images annotated with w_i is to the set of images annotated with w_j. The value of zero indicates no correlation and it means the two tags have not co-occurred in the past. The value of 1 indicated a strong correlation, meaning the set of images annotated with w_i is identical to that of w_j and the two tags have never appeared separately in the past.

5.2 Indirect Correlation

We can extend the notion of direct correlation to that of indirect correlation. Observe that even when two words may never have co-occurred together in any image, they could still be correlated to each other through other words. For instance, the words *beach* and *ocean* may have never been used in the same image as tags. However, if *beach* is seen often with the word *sand* and *sand* is often seen with the word *ocean*, then intuitively, the words *beach* and *ocean* are correlated to each other through the word *sand*.

To define indirect correlations, we apply the mathematical apparatus similar to the one developed for the diffusion kernels on graph nodes [30]. Specifically, we can define a *base correlation graph* as a graph $\mathcal{G} = (V, E)$ whose nodes are tags in the vocabulary V. An edge is created per each pair of nodes w_i and w_j and labeled with the value of $c(w_i, w_j)$. The *base correlation matrix* $B = B_1$ of $\mathcal{G}$ is a $V \times V$ matrix with elements $B_{ij} = c(w_i, w_j)$. Let P_{ij}^2 be the set of all paths of length two in graph $\mathcal{G}$ from w_i to w_j. Then, the indirect correlations $c_2(w_i, w_j)$ of length two for w_i and w_j is defined as the sum of contributions of each path $(x_0 x_1 x_2) \in P_{ij}^2$, where the contribution of each path is computed as the product of base similarities on its edges:

$$c_2(w_i, w_j) = \sum_{(x_0 x_1 x_2) \in P_{ij}^2} \prod_{i=1}^{2} c(x_{i-1}, x_i). \tag{5}$$

It can be shown that the corresponding similarity matrix B_2 can be computed as $B_2 = B^2$. The idea can be extended further by considering $c_k(w_i, w_j)$ and demonstrating that $B_k = B^k$. To take into account all of these indirect similarities for $k = 1, 2, \ldots, m$, the algorithm computes similarity matrix A in a manner similar to that of diffusion kernels. For instance, in spirit of the exponential diffusion kernel, A can be computed as $A = \sum_{k=0}^{m} \frac{1}{k!} \lambda^k B^k$, or, similar to the von Neumann diffusion kernel, as $A = \sum_{k=0}^{m} \lambda^k B^k$. From the efficiency perspective, it should be noted that computations of A and B_k for $k = 1, 2, \ldots, m$ are performed before processing of image annotations starts. Therefore, very fast computation of A is not critical. From practical perspective, however, there are known optimizations that significantly speed up these computation by employing eigen-decomposition of B, see [30] for details.

```
COMPUTE-ANSWER-BB (ℒ)
1  W* ← ∅    // Best sequence
2  s* ← 0    // Best score
3  R ← GET-M-BEST-SEQUENCES (ℒ, M, N_leaf)
4  for each W ∈ R do
5      s ← GET-SCORE(W)
6      if s > s* do
7          W* ← W
8          s* ← s
9  W* ← PICK-NULLS(W*)
10 return W*
```

Fig. 4. Overall algorithm with branch and bound.

5.3 Correlation and Membership Scores

Using the notion of correlation, we can define the *correlation score* $C(w)$ of sequence W as

$$C(W) = \sum_{w_i, w_j \in W, \text{ s.t. } i<j} A(w_i, w_j). \quad (6)$$

Its purpose is to assign higher values to sequences wherein the combinations of tags are more correlated. For direct correlations, this means the tags have co-occurred together more frequently in the past. Depending on whether direct or indirect correlations are used, the score can also be direct or indirect correlation score.

The *membership score* $M(W)$ of sequence W is computed as

$$M(W) = \sum_{w_i \in W} \frac{n(w_i)}{N_I}. \quad (7)$$

It reflects how often each tag w_i in W has been used in the past. Thus, it would assign a higher score to a combination of tags that have been more frequent in the past.

The correlation and membership scores $C(W)$ and $M(W)$ of sequence W can be combined as a linear combination into the *CM score* of the sequence:

$$S_{CM}(W) = \alpha C(W) + (1-\alpha) M(W). \quad (8)$$

The parameter α takes values in $[0, 1]$ interval and controls the relative contribution of the correlation and membership scores.

6 Branch and Bound Algorithm for Fast Searching of the Best Sequences

In this section, we discuss methods for speeding up the overall algorithm illustrated in Fig. 1. This algorithm has two main parts that can be optimized in terms of improving its efficiency and scalability:

1. *Sequence level.* Computing the score $S(W)$ for a given sequence W.
2. *Enumeration level.* Enumerating N^K sequences.

We have designed optimization techniques for both sequence and enumeration levels. For the ME score $S_{ME}(W)$, the intuition behind the sequence-level optimization is that computing $P(w_1, w_2, \ldots, w_K)$ using the ME has high computational complexity as a function of K. The idea is to split this computation into several computations for smaller values of K, by identifying the independent components for $w_1, w_2, \ldots, w_K$. The independent components are found using a clique finding algorithm applied to $\mathcal{G}_{ind}$. Several sequence-level optimization have also been proposed in [23]. In the subsequent discussion, we will focus mainly on an enumeration-level optimization.

```
GET-M-BEST-SEQUENCES(ℒ, M, N_leaf)
1  v ← NEW-NODE()  // Root Node
2  W_v ← (−, −, …, −)
3  ℓ_v ← 0, h_v ← ∞
4  N_ℓ ← N_leaf
5  R ← ∅    // Result Set
6  PUT(Q, v)  // Priority Queue
7  while NotEmpty(Q) do
8      ℓ* ← UPDATE-BOUND(Q, R)  // M-th-best lower bound
9      v ← GET(Q)
10     if h_v < ℓ* continue // Pruning
11     if |W_v| = K do
12         ADD-TO-RESULTS(R, W_v, M)
13         N_ℓ ← N_ℓ − 1
14         if N_ℓ ≤ 0 break
15         continue
16     L ← GET-LIST-TO-BRANCH(v)
17     BRANCH(L)
18 return R
```

Fig. 5. Branch and bound algorithm.

```
ADD-TO-RESULTS (R, W_v, M)
1  if |R| ≥ M do
2      u ← argmin_{u: W_u ∈ R} s_u
3      if s_u ≥ S(W_v) do
4          return
5      R ← R \ {W_u}  // Delete the worst element
6  R ← R ∪ {W_v}
7  return
```

Fig. 6. Updating the result set.

Naïvely enumerating all possible N^K sequences of words and finding a sequence with the best score, as outlined in Fig. 1, is a prohibitively expensive operation in terms of execution time. This section describes a faster algorithm for achieving the same goal, which trades efficiency for quality. The algorithm is designed to work with both the ME score function $S_{ME}(W)$ from Section 4 as well as the CM score function $S_{CM}(W)$ from Section 5. Virtually, all of the strategies of the algorithm are motivated by the need to find the next-best sequence as quickly as possible and in an incremental fashion.

Overall algorithm. If the CM score $S_{CM}(W)$ is used for scoring sequences, that is when $S(W) = S_{CM}(W)$, then the algorithm simply needs to invoke the Branch and Bound (BB) method shown in Figs. 5 and 6 with $M = 1$. This will return the desired top sequence W^* according to the CM score. The case of the ME score, that is when $S(W) = S_{ME}(W)$, is more challenging. The new overall algorithm for that case is illustrated in Fig. 4. Instead of performing a simple enumeration, it first invokes the Branch and Bound method shown in Fig. 5. The BB algorithm, given two parameters $M, N_{leaf} : 1 \leq M, N_{leaf} \leq N^K$, is capable of quickly computing the M high-score sequences according to some *indirect score*, which will be explained in detail in the subsequent discussion. The BB algorithm discovers sequences in a greedy fashion such that the best (higher score) sequences tend to be discovered first and lower score sequences—last. It maintains the set R of M highest score sequences observed thus far. The algorithm stops either 1) after examining N_{leaf}

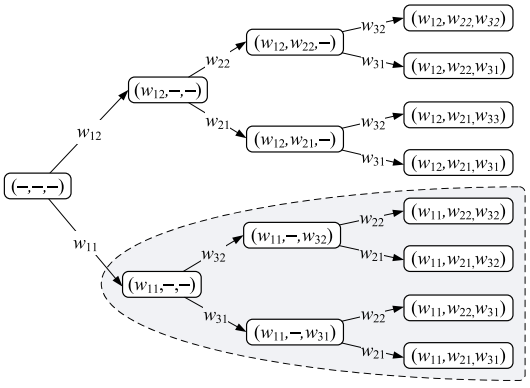


Fig. 7. Search tree.

discovered sequences or 2) if the current M sequences in R are guaranteed to be the top M sequences according to the indirect score.[1] After the invocation of the BB algorithm, the new overall algorithm then enumerates only among these few M sequences returned by BB and picks sequence W^* with the top ME scores. It then applies the null choosing procedure to W^* and outputs the resulting sequence as its final answer.

Indirect score. For the case of ME score, that is when $S(W) = S_{ME}(W)$, we will refer to $S_{ME}(W)$ as the *direct score* $S_{dir}(W) = S_{ME}(W)$, since it is the score we are interested in. We now will define a complementary *indirect score* function $S_{ind}(W)$, which is a function that should satisfy the following requirements:

- Function $S_{ind}(W)$ should behave similar to the direct score function $S_{dir}(W)$. Specifically, if for any two sequences W_1 and W_2 it holds that $S_{dir}(W_1) > S_{dir}(W_2)$ according to the direct score $S_{dir}(\cdot)$, then it should be likely that $S_{ind}(W_1) > S_{ind}(W_2)$ according to the indirect score $S_{ind}(\cdot)$.
- Even though the indirect function $S_{ind}(W)$ might not be as accurate in predicting the right sequence, it should be significantly faster to compute than the direct score function $S_{dir}(W)$.
- The functional form of the indirect function $S_{ind}(W)$ should allow computations of good upper and lower bounds, whose purpose will be explained shortly.

Choosing $S_{ind}(W) = S_{CM}(W)$ satisfies the desired requirements. As we will see from Section 8, $S_{CM}(W)$ tends to behave similar to the direct score function. Additionally, it is much faster to compute and can be very effectively bounded.

Search tree. The BB algorithm operates by traversing a search tree. Fig. 7 demonstrates an example of such a complete search tree where the number of lists K is three and each list contains $N = 2$ words. The tree is constructed on the fly as the algorithm proceeds forward. The algorithm might never visit certain branches of the tree, in which case they will not be constructed. A node u in the tree represents a sequence of words W_u. The sequences are grown by one word at a time during branching of nodes, starting from the root node sequence $(-, -, \ldots, -)$. A directed edge $u \rightarrow v$ with label w_{ij} from nodes u to v represents the fact that the sequence W_v of node v is obtained by adding word w_{ij} from list L_i to the sequence W_u of word u. That is, $W_v = W_u \cup \{w_{ij}\}$, which means for each kth element $W_v[k]$ of W_v, where $1 \leq k \leq K$, if $k \neq i$ then $W_v[k] = W_u[k]$, and if $k = i$ then $W_v[k] = W_v[i] = w_{ij}$.

1. As we will see in Section 8, in practice, optimal results in terms of quality and efficiency are obtained when $N_{leaf} \ll N^K$, and thus the stopping condition 2) rarely activates.

```
BRANCH(v, L)
1  for each w ∈ L do
2      u ← NEW-NODE()
3      W_u ← W_v ∪ {w}
4      m_u ← m_v + M(w) // Membership score
5      c_u ← GET-INCR-COREL(v, w) // Incremental from v
6      (ℓ_u, C_2v) ← GET-LOWER-BOUND(u)
7      h_u ← GET-UPPER-BOUND(u, C_2v)
8      PUT(Q, u)
```

Fig. 8. Branching procedure.

Priority queue. The algorithm maintains a priority queue Q for picking a node v that the algorithm considers to be the most promising sequence W_v to continue with. That is, W_v has the largest *chance* that it can be grown (by adding more words to W_v) into the best (highest score) sequence W^*. Initially, Q consists of only the root node. The choice of the key for the priority queue will be explained shortly when we discuss the bounding procedure. Intuitively, the value of the key should reflect the above-mentioned chance.

Branching. After picking the top node v from the priority queue Q, the algorithm performs a branching as follows (see Fig. 8): Let $\mathcal{L}$ be the set of all N-best lists. If sequence W_v contains a word taken from list L_j, then W_v is said to *cover* list L_j. Let $\mathcal{L}_v \subset \mathcal{L}$ be the set of lists that are already covered by W_v. The algorithm examines the elements of the lists $L_i \in \overline{\mathcal{L}}_v$ (where $\overline{\mathcal{L}}_v = \mathcal{L} \setminus \mathcal{L}_v$) that are not yet covered by W_v. Among them, it finds a word w that it considers to be the best to add to W_v next. It then perform a branching of v by adding N new nodes to the tree. Let L_i be the list wherein word w was found. Then, each of the new nodes corresponds to sequence $W_v \cup \{w_{ij}\}$, where $w_{ij} \in L_i$ for $j = 1, 2, \ldots, N$. That is, each sequence is formed by adding one word from list L_i to the current W_v sequence. Then, the lower and upper bounds are computed for the new N nodes (as explained shortly) and the N nodes are then inserted into the priority queue Q.

Choosing the N-best list to branch next. The algorithm uses two criteria for choosing the best N-best list to branch next, depending on whether or not it currently examines a root node, as demonstrated in Fig. 9. For the root node, the algorithm finds word w_{ij} such that the score $S(\{w_{ij}\} \cup \{w_{mn}\})$ is maximized over all possible $1 \leq i, m \leq K$ and $1 \leq j, n \leq N$, where $i \neq m$. If there are multiple such words, it then considers among them only those words that maximize $S(w_{ij})$, and picks one of them (if there are more than one) randomly. The N-best list that contains this word is picked for branching. For a nonroot node v, it scans through the elements of the lists that are not yet covered by W_v and picks word w_{ij} that maximizes score $S(W_v \cup \{w_{ij}\})$. Similarly, the list that contains this word is chosen for branching next.

Bounding. Let us revisit the issue of choosing the key in priority queue Q. Among the nodes in the priority

```
GET-LIST-TO-BRANCH(v)
1  M1 ← 0, M2 ← 0 // max scores
2  i* ← 1 // index of the best list
3  if v = root do
4     for i ← 1 to K − 1 do
5        for j ← 1 to N do
6           for m ← i + 1 to K do
7              for n ← 1 to N do
8                 m1 ← S({wij} ∪ {wmn})
9                 if m1 < M1 continue
10                m2 ← max(S(wij), S(wmn))
11                if m1 = M1 and m2 ≤ M2 continue
12                M1 ← m1, M2 ← m2
13                if S(wij) ≥ S(wmn) then i* ← i
14                else i* ← m
15    return Li*
    //– Handle non root nodes –
16  for each Li ∈ L̄v do
17     for each w ∈ Li do
18        if S(Wv ∪ w}) > M1 do
19           M1 ← S(Wv ∪ {w})
20           i* ← i
21  return Li*
```

Fig. 9. Choosing list for branching.

```
GET-LOWER-BOUND (v)
1  W_v^minS ← Wv
2  W_v^minC ← Wv // This var is for the upper bound
3  for each Li ∈ L̄v do
4     j ← argmax_n S(Wv ∪ {win})
5     W_v^minS ← W_v^minS ∪ {wij}
6     j ← argmax_n C(Wv ∪ {win})
7     W_v^minC ← W_v^minC ∪ {wij}
8  return (S(W_v^minS), C(W_v^minC))
```

Fig. 10. Computing lower bound.

```
GET-UPPER-BOUND (v, C2v)
1  M ← mv // Bounding membership
2  for each Li ∈ L̄v do
3     M ← M + M(Li)
4  C1v ← cv // Bounding correlations
5  C3v ← 0
6  for each Li ∈ L̄v do
7     for each Lj ∈ L̄v s.t. j > i do
8        C3v ← C3v + C(Li, Lj)
9  return M + C1v + C2v + C3v
```

Fig. 11. Computing upper bound.

queue, which one should we choose to branch next? Let us consider sequence W_v that corresponds to a node $v \in Q$. Let's first consider the case where $|W_v| < K$. For v, we can define the set of derivable answer sequences $D_v = \{W : W_v \rightsquigarrow W \text{ and } |W| = K\}$ wherein each sequence W is of size $|W| = K$ and is derivable from node v via the branching procedure described above. Such $2^{K-|W_v|}$ sequences correspond to the leaf nodes of the subtree of the complete search tree rooted at node v. Then, for node v, let s_v be the value of the maximum score among these sequences $s_v = \max_{W \in D_v} S(W)$. Notice that s_v would be an ideal key for the priority queue Q, as it would lead to the quickest way to find the best sequences. The problem is that the exact value of s_v is unknown when v is branched, since the sequences in D_v are not yet constructed at that moment.

Even though s_v is unknown, it is possible to quickly determine good lower and upper bounds on its value $\ell_v \le s_v \le h_v$, without comparing scores of each sequence in D_v. For the root node v, the bounds are computed as $\ell_v = 0$ and $h_v = \infty$. For any nonroot node v, if $|W_v| = K$, then the bounds are equal to the score of the sequence $\ell_v = h_v = S(W_v)$. If $|W_v| < K$, then the bounds are computed as explained next.

Lower bound. Given that $s_v = \max_{W \in D_v} S(W)$, to compute a lower bound on s_v, it is sufficient to pick one sequence W_v^{minS} from D_v and then set $\ell_v = S(W_v^{minS})$. The procedure for choosing such $W_v^{minS} \in D_v$ determines the quality of the lower bound. Specifically, the higher the score $S(W_v^{minS})$ of the chosen sequence, the tighter the value of ℓ_v is going to be. To pick a good W_v^{minS}, the proposed algorithm employs a strategy that examines the lists that are not yet covered by W_v, see Fig. 10. In each such list L_i, it finds the word w_{ij} that maximizes the score of a sequence constructed by adding a word from L_i to W_v: $j = \text{argmax}_n S(W_v \cup \{w_{in}\})$. Such words w_{ij} are then added to W_v to form W_v^{minS}.

Upper bound. To compute h_v, observe that the score $S(W)$ of sequence W is computed as a monotonic function of the correlation and memberships scores $C(W)$ and $M(W)$. Consequently, an upper bound on $\max_{W \in D_v} S(W)$ can be computed from upper bounds on $\max_{W \in D_v} C(W)$ and $\max_{W \in D_v} M(W)$, as demonstrated in Fig. 11. To bound $\max_{W \in D_v} M(W)$, observe that we can precompute beforehand for each list $L_i \in \mathcal{L}$, the maximum membership score $M(L_i)$ reachable on words from that list: $M(L_i) = \max_{w_{ij} \in L_i} M(w_{ij})$. Therefore, $\max_{W \in D_v} M(W)$ can be upper bounded by $\sum_{w \in W_v} M(w) + \sum_{L \in \overline{\mathcal{L}}_v} M(L)$.

To bound $\max_{W \in D_v} C(W)$, observe that this maximum will be reached on some yet unknown sequence W_v^{max} derived from W_v, that is $\max_{W \in D_v} C(W) = C(W_v^{max})$. Observe that $C(W_v^{max})$ is computed as a sum of correlations among distinct pairs of distinct words $w', w'' \in W_v^{max}$, that is, $C(W_v^{max}) = \sum_{w', w'' \in W_v^{max}} A(w', w'')$. This computation can be logically divided into three parts, based on whether a word is taken from W_v or from the rest of the words in W_v^{max}:

1. $\sum_{w', w'' \in W_v} A(w', w'')$,
2. $\sum_{w' \in W_v, w'' \in W_v^{max} \setminus W_v} A(w', w'')$,
3. $\sum_{w', w'' \in W_v^{max} \setminus W_v} A(w', w'')$.

We now will explain how to bound $C(W_v^{max})$ by specifying how to bound each of its three parts. The sum of correlations from the first category c_v is known exactly. It is computed once and stored in node v by the algorithm to avoid unnecessary recomputations during branching of node v, as shown in Fig. 8. The bound on the correlations from the second category can be found during the computation of the lower bound in a similar fashion. Namely, in addition to looking for w_{ij} word in list L_i that maximizes the score, the algorithm also looks for $w'_{ij} \in L_i$ word that maximized the correlation score from words of W_v to words from L_i. By construction, the sum of correlations from the second category cannot exceed the sum of correlations from word in W_v to such w'_{ij} words. Finally, the correlation of the third category where two words come from $L_i, L_j \in \overline{\mathcal{L}}_v$ can

be bounded by the maximum correlation that exist between a word from L_i and a word from L_j. Such maximum correlations are computed once on the need basis and then stored in a table to avoid their recomputations.

The values of lower and upper bound can be used to estimate s_v. Such estimation can serve as the key for the priority queue. Specifically, we employ the lower bound as the key, as it tends to be a tighter bound. The lower and upper bound are also utilized to prune the search space, as explained below.

Pruning. The algorithm maintains the Mth best guaranteed lower bound ℓ^* observed thus far. Its value means that the algorithm can guarantee that it can find M sequences such that the minimum lower bound among them will be greater or equal than ℓ^*. Initially, the value of ℓ^* is set to 0. Observe that if for some node v it holds that $h_v < \ell^*$ then the entire subtree rooted at v can be pruned away from further consideration. This is because none of the sequences that correspond to the leaf-level nodes of this subtree can reach a score higher than ℓ^*, whereas a score of at least ℓ^* is reachable by M sequences of other nonoverlapping subtrees.

Discussion. The BB algorithm creates a trade-off between the execution time and quality of the result. That is, the larger the values of M and N_{leaf} the slower the algorithm becomes, but the better the quality the algorithm gets, until the quality reaches a plateau.

7 Extensions of Framework

7.1 Detecting Nulls

This section discusses how correlations can be utilized for detecting null candidates. That is, detecting the situation that a given N-best list L_i is unlikely to contain the ground truth tag g_i.

First, we extend the notion of a base correlation graph $\mathcal{G}$ to that of indirect correlation graph $\mathcal{G}_{ind}$. Like in $\mathcal{G}$, the nodes of $\mathcal{G}_{ind}$ are the tags $w_i \in V$, but each edge (w_i, w_j) is now labeled with the value of A_{ij}.

Let $W^* = (w_1, w_2, \ldots, w_K)$ be the sequence with the highest score among all the possible N^K sequences for a given sequence of N-best lists $\mathcal{L}$. If list $L_i \in \mathcal{L}$ does not contain the ground truth tag g_i, then $w_i \neq g_i$. We can observe that when such situations occur, it is likely that w_i will not be strongly correlated with the rest of the tags in W^*.

Given these two observations, we can design the null detection procedure. It takes $W^* = (w_1, w_2, \ldots, w_K)$ as input and analyzes each $w_i \in W^*$. If $A(w_i, w_j) < \tau$ for $j = 1, 2, \ldots, K, j \neq i$, and a threshold value τ, then w_i is considered to be *isolated* in $\mathcal{G}_{ind}$, in terms of correlations, from the rest of the tags. Isolated tags are then substituted with `null` values.

7.2 Combining Results of Multiple Models

So far our discussion has focused on how correlation semantics derived from a corpus of images can be used to improve the annotation quality. We refer to this collection of all images published by all users as the global corpus. In addition to tag correlations in the global corpus, we may further be able to exploit local information in local collections of users, e.g., the set of images belonging to the user, the information in calendars, e-mails, and so on. Adding additional semantics can be achieved by training a local model based on the local content belonging to the user. While the local semantics may take multiple forms, we restrict ourselves to correlation semantics only. We now have a challenge of two models—a local model based on the local collection belonging to the user, and a global model which is aggregated over multiple users. We can combine the two models to further improve recognition effectiveness.

We will primarily focus on two scenarios: 1) the global model and 2) global and local model for CM score $S_{CM}(W)$. The global model scenario is a single-model scenario. It assigns scores to sequences based on how all of the users tagged images in the past in general. The local model for a particular user, instead of being applied to the entire corpus of images $\mathcal{D}_G$, is applied to only the set of images $\mathcal{D}_L$ of this user. As such, the local model is tuned to a specific user in terms of his vocabulary V_L and the way he tags images. Thus, combining the global and local models has the potential for improving the quality of annotation for a specific user.

Suppose that for a user his local profile is available. Then, we can apply the global and local models $\mathcal{M}_G$ and $\mathcal{M}_L$ to score a sequence $W = (w_1, w_2, \ldots, w_K)$ generated by the user. Namely, the overall score $S(W)$ is computed as a linear combination of the global and local scores: $S(W) = \gamma S_G(W) + (1 - \gamma) S_L(W)$. Here, $S_G(W)$ and $S_L(W)$ are computed as $S(W)$ for the single-model case, except for $S_G(W)$ is computed over the global corpus of images whereas $S_L(W)$ over the local corpus, specific to the user. The parameter $\gamma \in [0, 1]$ controls the relative contribution of each score.

8 Experiments

In this section, we empirically evaluate the proposed approach in terms of both the quality and efficiency on real and synthetic data sets.

Data sets. We test the proposed approach on three data sets. The data sets have been generated by web crawling of a popular image hosting website Flickr.

1. `Global` is a data set consisting of 60,000 Flickr images. We randomly set aside 20 percent of the data for testing (will be called G_{test}) and 80 percent for training (G_{train}). We will use portions of G_{test} for testing, e.g., 500 random images. The size of the global vocabulary is $|V_G| = 18{,}285$. Since it is infeasible to provide speech annotations for a large collection of images, the N-best list for this data set have been generated synthetically. Namely, we use the Metaphone algorithm to generate three to four alternatives for the ground truth tags. We also have used parameters to control the uncertainty of the data: the probability that an N-best list will contain the ground truth tag.
2. `Local` is a data set consisting of images of 65 randomly picked prolific picture takers (at least 100 distinct tags and 100 distinct images). For each user, we randomly set aside 20 percent of the data for testing (L_{test}) and use various portions of the remaining 80 percent for training (L_{train}) the local model. The Metaphone algorithm is employed to generate alternatives for the ground truth tags in L_{test}.
3. `Real` is a real data set generated by picking 100 images from G_{test} and annotating them (generating the N-best

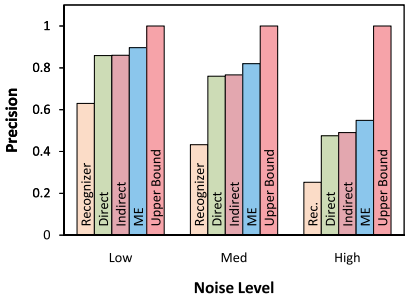


Fig. 12. Precision versus noise.

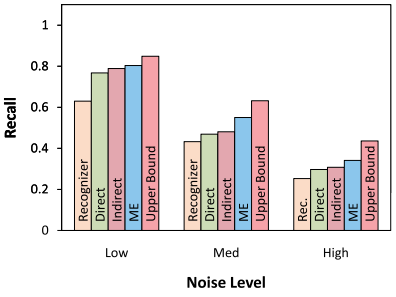


Fig. 13. Recall versus noise.

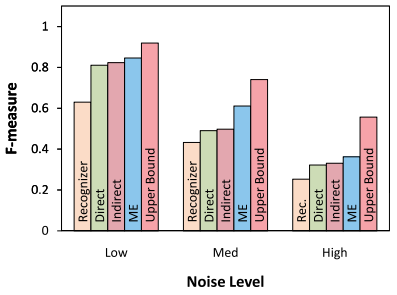


Fig. 14. F-measure versus noise.

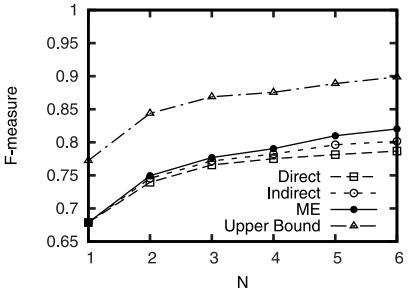


Fig. 15. F-measure versus N.

lists) using a popular commercial off-the-shelf recognizer Dragon v.8. The annotations were performed in a *Low* noise level. Low noise level corresponds to a quiet university lab environment. All non-English words were removed before using *Dragon* to create these N best lists.

Approaches. We will compare the results of five approaches:

- `Recognizer` is the output of the recognizer (Dragon v.8).
- `Direct` is the proposed solution with S_{CM} score and Direct Correlation (Section 5.1).
- `Indirect` is the proposed solution with S_{CM} score and Indirect Correlation (Section 5.2).
- `ME` is the proposed solution with S_{ME} score, that is based on Max Entropy (Section 4).
- `Upper Bound` is the theoretic upper bound achievable by the class of the algorithms we consider (Section 3).

As we will see in the rest of this section, `ME` tends to be the best approach in terms of quality but at the cost of lower efficiency. `Direct` and `Indirect` methods get lower quality but significantly more efficient. The choice of the approach thus can be dictated by specific needs of the underlying application—whether it requires quality over efficiency or vice versa.

Experiment 1: (Quality for various noise levels). We randomly picked 20 images from `Real` and created N-best lists for their annotations using *Dragon* in two additional noise levels: *Medium* and *High*. Medium and High levels have been produced by introducing white Gaussian noise through a speaker.[2] Figs. 12, 13, and 14 study the Precision, Recall, and F-measure of all the approaches for the Low, Medium, and High noise levels on these 20 images.

2. To give a sense of the level of noise, High was a little louder than the typical volume of TV in a living room.

Since we created `Real` in a Low noise level on 100 images, for a fair comparison, the points corresponding to Low noise levels in the plots are averages over these 20 images, as opposed to the all 100 images. As anticipated, higher noise levels negatively affect performance of all the approaches. In this experiment, the results are consistent in terms of precision, recall, and F-measure: at the bottom is `Recognizer`, then `Direct`, then `Indirect`, followed by `ME`, and then by Upper Bound. As expected, `Indirect` is slightly better than `Direct`. In turn, `ME` tends to dominate `Indirect`. `ME` consistently outperforms `Recognizer` by 11-22 percent of F-measure across the noise levels and it is also within 7-20 percent of F-measure from `Upper Bound`. In the subsequent discussion, we will refer to `Real` data with the Low level of noise as just `Real`.

Experiment 2: (Quality versus size of N-Best lists). Fig. 15 illustrates the F-measure as a function of N (the size of the N-best list) on `Real` data. For a given N, the N-best lists are generated by taking the original N-best lists from `Real` data and keeping at most N first elements in them. Increasing N presents a trade-off. Namely, as N increases, the greater is the chance that the ground truth element will appear in the list. At the same time, `Direct`, `Indirect`, and `ME` algorithms are faced with more uncertainty as there will be more options to disambiguate among. The results demonstrate that the potential benefit from the former outweighs the potential loss due to the latter, as the F-measure increases with N. As expected, the results of `Indirect` are slightly better than those of `Direct`. As in the previous experiment, `ME` tends to outperform `Indirect`.

Experiment 3: (Correlation of direct and indirect scores). Section 6 has discussed that one of the requirements for the indirect score function is that it should behave similar to the direct score function. Fig. 16 demonstrates the correlation between the two scoring functions. It plots Hit Ratio as a function of Best M, which is the probability that the top sequence according to the direct score is contained within the best M sequences according to the indirect score

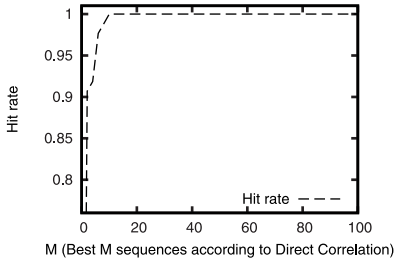


Fig. 16. Similarity of direct and indirect scores.

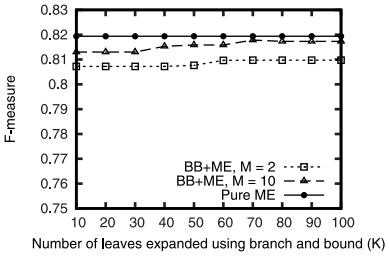


Fig. 17. Quality of BB algorithm.

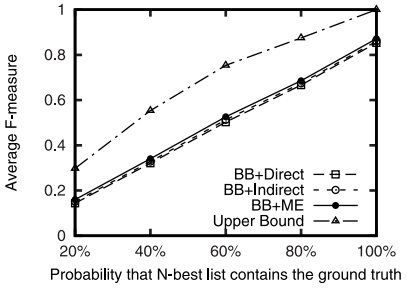


Fig. 18. Quality on a larger data set.

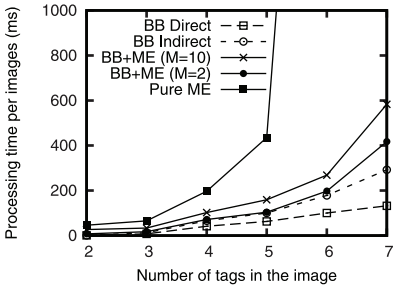


Fig. 19. Processing time.

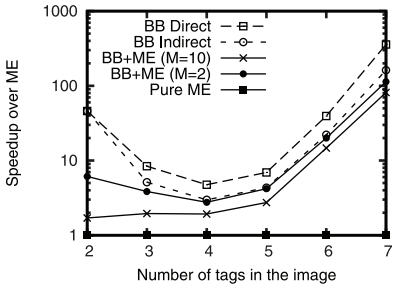


Fig. 20. Speedup of BB algorithm.

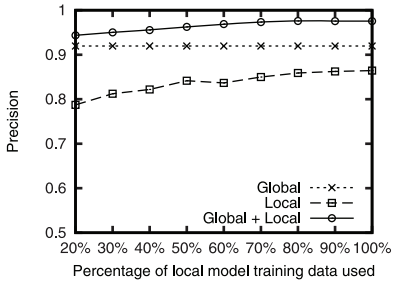


Fig. 21. Multimodel test: precision.

on Real data set. The figure demonstrates that the two chosen scoring functions are indeed very closely correlated, as it is necessary to consider very few best indirect score sequences to get the top direct score sequence.

Experiment 4: (Quality of the branch and bound algorithm). The goal of the Branch and Bound algorithm is to reduce the exponential search space of the problem. Once a substantial portion of the search space is pruned, the algorithm enumerates over the potentially best M sequences in the unpruned space and picks a single sequence that is assigned the highest score by Maximum Entropy. The Branch and Bound algorithm stops after observing N_{leaf} leaf nodes of the search tree. Fig. 17 demonstrates the effect of N_{leaf} on the F-measure of the BB algorithm for $M = 2$ and $M = 10$ on Real data set. We can observe that the quality increases as N_{leaf} increases. After expanding only about $N_{leaf} = 60$ leaf nodes, the algorithm is within 1 percent of the overall F-measure. While we do not plot Precision and Recall for space constraints, similar trends are observed.

Experiment 5: (Quality on a larger data set). Fig. 18 studies the F-measure of the proposed approach (with BB algorithm) on a larger Global data set. We set $M = 15$ and $N_{leaf} = 175$ and vary P_{truth} (the probability that a given list will contain the ground truth tag) from 20 to 100 percent. The figure demonstrates that as P_{truth} increases the quality of the algorithm increases as well, as more correct tags become available for analysis. Similar trends are observed in the case of Precision and Recall. Again, as expected, the results of Indirect are marginally better than those of Direct. The results of ME are slightly better than those of Indirect.

Experiment 6: (Speedup of branch and bound). This experiment studies the efficiency of the branch and bound method. Fig. 19 plots the overall time, in milliseconds, taken per image for various techniques. Fig. 20 shows the speedup achieved by the same techniques over the pure ME algorithm, which is computed as the time of the ME divided by the time of the technique. The figures demonstrate that Direct is the most effective solution, followed by Indirect, then by BB+ME with $M = 2$ and $M = 10$, $N_{leaf} = 75$. The figures illustrate that as the number of tags in image annotations increases, the speedup rapidly increases as well, reaching two orders of magnitude for $K = 7$. For small values of K, the processing time of our implementation of pure ME increases slower than that of the other techniques. This causes the speedup for the other techniques to decrease initially, but then grow again as K increases, reaching roughly 360 times for Direct.

Experiment 7: (Local profile and multimodel). In this experiment, we evaluate the quality of the Local, Global, and Global+Local models explained in Section 7.2. The tests are performed on Local data set. Figs. 21, 22, and 23 demonstrate the Precision, Recall, and F-measure achieved by the three models.

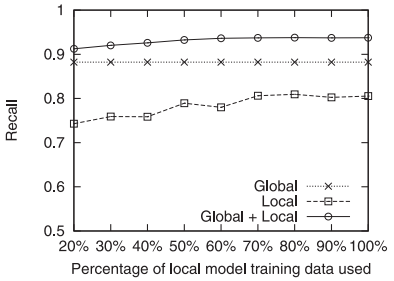


Fig. 22. Multimodel test: recall.

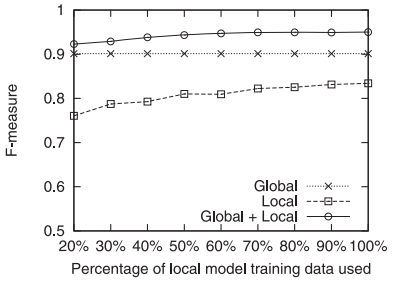


Fig. 23. Multimodel test: F-measure.

Recall that in `Local`, 80 percent of the data has been set aside for training. The X-axis in each of these figures plots the percentage of these 80 percent that have been used for training. The performance of the Global model does not change with the size of the training set used for the Local model. As expected, the performance of the purely Local model increases with the size of the training set used for learning. The multimodel approach Global+Local dominates both, the Global and Local model. The reason for it is that like the Local model it knows which combination of words the user prefers. In addition, it is also aware of common tag combinations from the global profile that are not present in the local (training) profile of the user. As anticipated, when the amount of training data is small, Global+Local behaves close to Global. The difference increases with the increase of the amount of training data available.

9 Conclusions and Future Work

This paper proposes an approach for using speech for text annotation of images. The proposed solution employs semantics captured in the form of correlations among image tags to better disambiguate between alternatives that the speech recognizer suggests. We show that semantics used in this fashion significantly improves the quality of recognition, which, in turn, leads to more accurate annotation. As future work, we plan to incorporate other sources of semantic information, including but not restricted to social network of the picture taker, the picture taker's address book, domain ontologies, visual properties of the image, etc.

Acknowledgments

This research was supported by the US National Science Foundation (NSF) Awards 0331707, 0331690, 0812693 and DHS Award EMW-2007-FP-02535.

References

[1] R. Bayeza-Yates and B. Riberto-Neto, *Modern Information Retrieval.* Addison-Wesley, 1999.

[2] D.M. Blei and M.I. Jordan, "Modeling Annotated Data," *Proc. ACM SIGIR,* 2003.

[3] J. Chen, T. Tan, and P. Mulhem, "A Method for Photograph Indexing Using Speech Annotation," *Proc. Second IEEE Pacific Rim Conf. Multimedia: Advances in Multimedia Information Processing (PCM),* 2001.

[4] S. Chen, D.V. Kalashnikov, and S. Mehrotra, "Adaptive Graphical Approach to Entity Resolution," *Proc. ACM/IEEE Joint Conf. Digital Libraries (JCDL),* June 2007.

[5] Z. Chen, D.V. Kalashnikov, and S. Mehrotra, "Exploiting Relationships for Object Consolidation," *Proc. ACM SIGMOD Workshop Information Quality in Information Systems (IQIS '05),* June 2005.

[6] Z.S. Chen, D.V. Kalashnikov, and S. Mehrotra, "Exploiting Context Analysis for Combining Multiple Entity Resolution Systems," *Proc. ACM SIGMOD,* June/July 2009.

[7] C. Desai, D.V. Kalashnikov, S. Mehrotra, and N. Venkatasubramanian, "Using Semantics for Speech Annotation of Images," *Proc. IEEE Int'l Conf. Data Eng. (ICDE),* Mar./Apr. 2009.

[8] O. Díaz, J. Iturrioz, and C. Arellano, "Facing Tagging Data Scattering," *Proc. Int'l Conf. Web Information Systems Eng. (WISE),* 2009.

[9] T. Hofmann, "Unsupervised Learning by Probabilistic Latent Semantic Analysis," *Machine Learning,* vol. 42, nos. 1/2, pp. 177-196, 2001.

[10] Y. Jin, L. Khan, L. Wang, and M. Awad, "Image Annotations by Combining Multiple Evidence & Wordnet," *Proc. ACM Int'l Conf. Multimedia,* pp. 706-715, 2005.

[11] D. Jurafsky and J. Martin, *Speech and Language Processing.* Prentice-Hall, 2000.

[12] D.V. Kalashnikov, Z. Chen, S. Mehrotra, and R. Nuray, "Web People Search via Connection Analysis," *IEEE Trans. Knowledge and Data Eng.,* vol. 20, no. 11, pp. 1550-1565, Nov. 2008.

[13] D.V. Kalashnikov, Z. Chen, R. Nuray-Turan, S. Mehrotra, and Z. Zhang, "WEST: Modern Technologies for Web People Search," *Proc. IEEE Int'l Conf. Data Eng. (ICDE),* demo publication, Mar./Apr. 2009.

[14] D.V. Kalashnikov and S. Mehrotra, "Domain-Independent Data Cleaning via Analysis of Entity-Relationship Graph," *ACM Trans. Database Systems,* vol. 31, no. 2, pp. 716-767, 2006.

[15] D.V. Kalashnikov, S. Mehrotra, S. Chen, R. Nuray, and N. Ashish, "Disambiguation Algorithm for People Search on the Web," *Proc. IEEE Int'l Conf. Data Eng. (ICDE),* short publication, 2007.

[16] D.V. Kalashnikov, S. Mehrotra, and Z. Chen, "Exploiting Relationships for Domain-Independent Data Cleaning," *Proc. SIAM Int'l Conf. Data Mining,* 2005.

[17] D.V. Kalashnikov, R. Nuray-Turan, and S. Mehrotra, "Towards Breaking the Quality Curse. A Web-Querying Approach to Web People Search," *Proc. ACM SIGIR,* July 2008.

[18] M.P. Kato, H. Ohshima, S. Oyama, and K. Tanaka, "Can Social Tagging Improve Web Image Search?" *Proc. Int'l Conf. Web Information Systems Eng. (WISE),* 2008.

[19] A. Kuchinsky, C. Pering, M.L. Creech, D.F. Freeze, B. Serra, and J. Gwizdka, "FotoFile: A Consumer Multimedia Organization and Retrieval System," *Proc. SIGCHI Conf. Human Factors in Computing Systems: The CHI is the Limit (CHI),* 1999.

[20] R. Lienhart, "A System for Effortless Content Annotation to Unfold the Semantics in Videos," *Proc. IEEE Workshop Content-Based Access of Image and Video Libraries (CBAIVL),* 2000.

[21] R.W. Lienhart, "Dynamic Video Summarization of Home Video," *Proc. SPIE Conf.,* 1999.

[22] C. Manning and H. Schutze, *Foundations of Statistical Natural Language Processing.* MIT Press, 1999.

[23] V. Markl, P.J. Haas, M. Kutsch, N. Megiddo, U. Srivastava, and T.M. Tran, "Consistent Selectivity Estimation via Maximum Entropy," *VLDB J.,* vol. 16, no. 1, pp. 55-76, 2007.

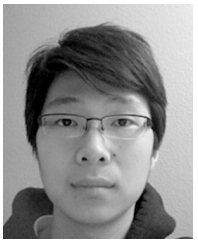
[24] F. Monay and D. Gatica-Perez, "On Image Auto-Annotation with Latent Space Models," *Proc. ACM Int'l Conf. Multimedia,* 2003.

[25] R. Nuray-Turan, Z. Chen, D.V. Kalashnikov, and S. Mehrotra, "Exploiting Web Querying for Web People Search in WePS2," *Proc. Second Web People Search Evaluation Workshop (WePS 2009), 18th Int'l World Wide Web (WWW) Conf.,* Apr. 2009.

[26] R. Nuray-Turan, D.V. Kalashnikov, and S. Mehrotra, "Self-Tuning in Graph-Based Reference Disambiguation," *Proc. 12th Int'l Conf. Database Systems for Advanced Applications (DASFAA),* Apr. 2007.
[27] S.D. Pietra, V.J.D. Pietra, and J.D. Lafferty, "Inducing Features of Random Fields," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 4, pp. 380-393, Apr. 1997.
[28] SAFIRE Project, http://www.ics.uci.edu/projects/cert/SAFIRE/, 2010.
[29] C.E. Shannon, *The Mathematical Theory of Communication.* Univ. of Illinois Press, 1949.
[30] J. Shawe-Taylor and N. Cristianni, *Kernel Methods for Pattern Analysis.* Cambridge Univ. Press, 2004.
[31] R.K. Srihari and Z. Zhang, "Show&Tell: A Semi-Automated Image Annotation System," *IEEE MultiMedia,* vol. 7, no. 3, pp. 61-71, July-Sept. 2000.
[32] A. Stent and A. Loui, "Using Event Segmentation to Improve Indexing of Consumer Photographs," *Proc. ACM SIGIR,* 2001.
[33] C. Wang, F. Jing, L. Zhang, and H. Zhang, "Image Annotation Refinement Using Random Walk with Restarts," *Proc. ACM Int'l Conf. Multimedia,* pp. 647-650, 2006.
[34] C. Wang, F. Jing, L. Zhang, and H.-J. Zhang, "Content-Based Image Annotation Refinement," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition (CVPR),* 2007.
[35] T. Watanabe, H. Tsukada, and H. Isozaki, "A Succinct N-Gram Language Model," *Proc. Joint Conf. 47th Ann. Meeting of the Assoc. for Computational Linguistics and Fourth Int'l Joint Conf. Natural Language Processing (ACL-IJCNLP),* 2000.

Dmitri V. Kalashnikov received the diploma (summa cum laude) in applied mathematics and computer science from Moscow State University, Russia, in 1999 and the PhD degree in computer science from Purdue University in 2003. Currently, he is an assistant adjunct professor at the University of California, Irvine. He has received several scholarships, awards, and honors, including an Intel Fellowship and Intel Scholarship. His current research interests are in the areas of entity resolution and disambiguation, web people search, spatial situational awareness, moving-object databases, spatial databases, and GIS.

Sharad Mehrotra received the PhD degree in computer science from the University of Texas at Austin in 1993. He is currently a professor in the Department of Computer Science at the University of California, Irvine (UCI) and the director of the Center for Emergency Response Technologies. Previously, he was a professor at the University of Illinois at Urbana-Champaign (UIUC). He has received numerous awards and honors, including SIGMOD Best Paper award 2001, DASFAA Best Paper award 2004, and CAREER award 1998 from the US National Science Foundation (NSF). His primary research interests are in the area of database management, distributed systems, and data analysis. He is a member of the IEEE.



Jie Xu received the BSc degree in computer science from Zhejiang University, China, in 2006 and the MSc degree in computer science from Zhejiang University, China, in 2008. He is currently a PhD candidate in the Computer Science Department of the University of California, Irvine. His research interests include information retrieval, machine learning, and computer vision.

Nalini Venkatasubramanian received the MS and PhD degrees in computer science from the University of Illinois in Urbana-Champaign. She is currently a professor in the School of Information and Computer Science at the University of California Irvine. She has had significant research and industry experience in the areas of distributed systems, adaptive middleware, mobile computing, distributed multimedia servers, formal methods and object-oriented databases. She is the recipient of the prestigious US National Science Foundation (NSF) Career award in 1999, an Undergraduate Teaching Excellence award from the University of California, Irvine in 2002 and multiple best paper awards. She is a member of the IEEE and the ACM, and has served extensively in the program and organizing committees of conferences on middleware, distributed systems and multimedia. Prior to arriving at UC Irvine, she was a research staff member at the Hewlett-Packard Laboratories in Palo Alto, California.