GCap: Graph-based Automatic Image Captioning*

Jia-Yu Pan Hyung-Jeong Yang Christos Faloutsos

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, U.S.A.

Pinar Duygulu

Department of Computer Engineering
Bilkent University
Ankara, Turkey, 06800

Abstract

Given an image, how do we automatically assign keywords to it? In this paper, we propose a novel, graph-based approach (GCap) which outperforms previously reported methods for automatic image captioning. Moreover, it is fast and scales well, with its training and testing time linear to the data set size. We report auto-captioning experiments on the "standard" Corel image database of 680 MBytes, where GCap outperforms recent, successful auto-captioning methods by up to 10 percentage points in captioning accuracy (50% relative improvement).

1. Introduction and related work

Given a huge image database, how do we assign content-descriptive keywords to each image, automatically? In this paper, we propose a novel, graph-based approach (GCap) which, when applied for the task of image captioning, outperforms previously reported methods.

Problem 1 (Auto-captioning) *Given a set $\mathcal{I}$ of color images, each with caption words; and given one more, uncaptioned image I_q ("query image"), find the best p (say, p=5) caption words to assign to it.*

Maron et al. [17] use multiple instance learning to train classifiers to identify particular keywords from image data using labeled bags of examples. In their approach, an image is an "positive" example if it contains a particular object (e.g. tiger) in the image, but "negative" if it doesn't. Wenyin et al. [26] propose a semi-automatic strategy for annotating images using the user's feedback of the retrieval system. The query keywords which receive positive feedback are collected as possible annotation to the retrieved images. Li and Wang [15] model image concepts by 2-D multiresolution Hidden Markov Models and label an image with the concepts best fit the content.

Recently, probabilistic models are proposed to capture the joint statistics between image regions and caption terms, for example, the co-occurrence model [19], latent semantic analysis (LSA) based models [18], machine translation model [3, 9], and the relevance-based language model [12]. These methods quantize or cluster the image features into discrete tokens and find correlations between these tokens and captioning terms. The quality of tokenization could effect the captioning accuracy.

Other work models directly the association between words and the numerical features of the regions, for example, the generative hierarchical aspect model [3, 4], the correspondence Latent Dirichlet Allocation [5], the continuous-space relevance model (CRM) [14], and the contextual model which models spatial consistency by Markov random field [7]. These methods try to find the actual association between image regions and terms for image annotation and for a greater goal of object recognition. In contrast, our proposed method GCap captions an entire image, rather than captioning by naming the constituent regions.

The focus of this paper is on auto-captioning. However, our proposed GCap method is in fact more general, capable of attacking the general problem of finding correlations between arbitrary modalities of arbitrary multimedia collections. In auto-captioning, it finds correlations between two modalities, image features and text. In a more general setting, say, of video clips, GCap can be easily extended to find correlations between some other modalities, like, e.g., the audio parts and the image parts. We elaborate on the generality of GCap later (subsection 2.4).

Section 2 describes our proposed method and its algorithms. In section 3 we give experiments on real data. We discuss our observations in section 4. Section 5 gives the

*This material is based upon work supported by the National Science Foundation under Grants No. IIS-0121641, IIS-9817496, IIS-9988876, IIS-0083148, IIS-0113089, IIS-0209107, IIS-0205224, INT-0318547, SENSOR-0329549, EF-0331657, IIS-0326322, by the Pennsylvania Infrastructure Technology Alliance (PITA) Grant No. 22-901-0001, and by the Defense Advanced Research Projects Agency under Contract No. N66001-00-1-8936. Additional funding was provided by donations from Intel, and by a gift from Northrop-Grumman Corporation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties.

conclusions.

2. Proposed Method

The main idea is to turn the image captioning problem into a graph problem. Next we describe (a) how to generate this graph, (b) how to caption a new image with this graph, and (c) how to do that efficiently.

Table 1 shows the symbols and terminology we used in the paper.

Symbol	Description
	Images/Objects
I	I_i: the i-th captioned image, I_q: the query image
$\mathcal{I}$	set of captioned images $\{I_1, \ldots, I_{N_I}\}$
$V(I_i)$	the vertex of GCap graph corresponding to image I_i.
$V(R_i)$	the vertex of GCap graph corresponding to region R_i.
$V(T_i)$	the vertex of GCap graph corresponding to term T_i.
k	the number of neighbors to be considered
c	the restart probability
	Sizes
N_I	the total number of captioned images
N_R,N_T	the total number of regions/terms from the captioned images
$N_R(I_q)$	the number of regions in the query image I_q
N	N=N_I+N_R+N_T+1+$N_R(I_q)$, the number of nodes in GCap graph
E	the number of edges in GCap graph
	Matrix/vector
$\mathbf{A}$	the (column-normalized) adjacency matrix
$\vec{\mathbf{v_q}}$	the restart vector (all zeros, except a single '1' at the element corresponding to the query image I_q)
$\vec{\mathbf{u_q}}$	the steady state probability vector with respect to the $\vec{\mathbf{v_q}}$
$u_s(v)$	the affinity of node "v" with respect to node "s"

Table 1: Summary of symbols used

The information about how image regions are associated with terms is established from a captioned image set. Each image in a captioned image set is annotated with terms describing the image content. Captioned images can come from many sources, for example, news agency [10] or museum websites. News agencies usually present pictures with good and concise captions. These captions usually contain the names of the major people, objects, and activities in a picture. Besides, images with high-quality captions are continually generated by human efforts [25].

We are given a set of captioned images $\mathcal{I}$, and an uncaptioned, query image I_q. Each captioned image has one or more caption words. For every image, we extract a set of feature vectors, one for each homogeneous region (a.k.a. "blob") of the image, to represent the content of an image, See Figure 1 for 3 sample images, their captions and their regions.

Thus, every captioned image has two *attributes*: (a) the caption (set valued, with strings as atomic values) and (b) the image regions (set valued, with feature vectors as atomic values).

We use a standard segmentation algorithm [23] to break an image into regions (see Figure 1(d,e,f)), and then map each region into a 30-d feature vector. We used features like the mean and standard deviation of its RGB values, average responses to various texture filters, its position in the entire image layout, and some shape descriptors (e.g., major orientation and the area ratio of the bounding region to the real region). All features are normalized to have zero-mean and unit-variance. Note that the exact feature extraction details are *orthogonal* to our approach - all our GCap method needs is a black box that will map each color image into a set of zero or more feature vectors to represent the image content.

How do we use the captioned images to caption the query image I_q? The problem is to capture the correlation between image features and caption terms. Should we use clustering or some classification methods to "tokenize" the numerical feature vectors, as it has been suggested before? And, if yes, how many cluster centers should we shoot for? Or, if we choose classification, which classifier should we use? Next, we show how to bypass all these issues by turning the task into a graph problem.

2.1 Graph-based captioning (GCap)

The main idea is to represent all the images, as well as their attributes (caption words and regions) as nodes and link them according to their known association into a graph. For the task of image captioning, we need a graph with 3 types of nodes. The graph is a "3-layer" graph, with one layer of image nodes, one layer of captioning term nodes, and one layer for the image regions. See Figure 1 for an example.

Graph construction We will denote as $V(I)$ the vertex of an image I, and as $V(T_i)$, $V(R_j)$ to be the vertex for the term T_i, and for the region R_j, respectively. There is one node for each image, one node for each distinct caption term, and one node for each region. Nodes are connected based on either (1) the co-occurrence relation or (2) the similarity relation.

To capture cross-attribute correlation, for each captioned image, we put edges between the image-node and the attribute-value nodes associated with the image. These edges are called the "image-attribute-value" links (IAV-links).

For the feature vectors of the regions, we need a way to reflect the similarity between them. For example, we would like to associate the orange regions r_6 and r_{10} which are both "tiger", to accommodate various appearances of the same object. Our approach is to add an edge if and only if the two feature vectors are "close enough". In our setting, we use the Euclidean distance between region feature vectors to denote (dis-)similarity.

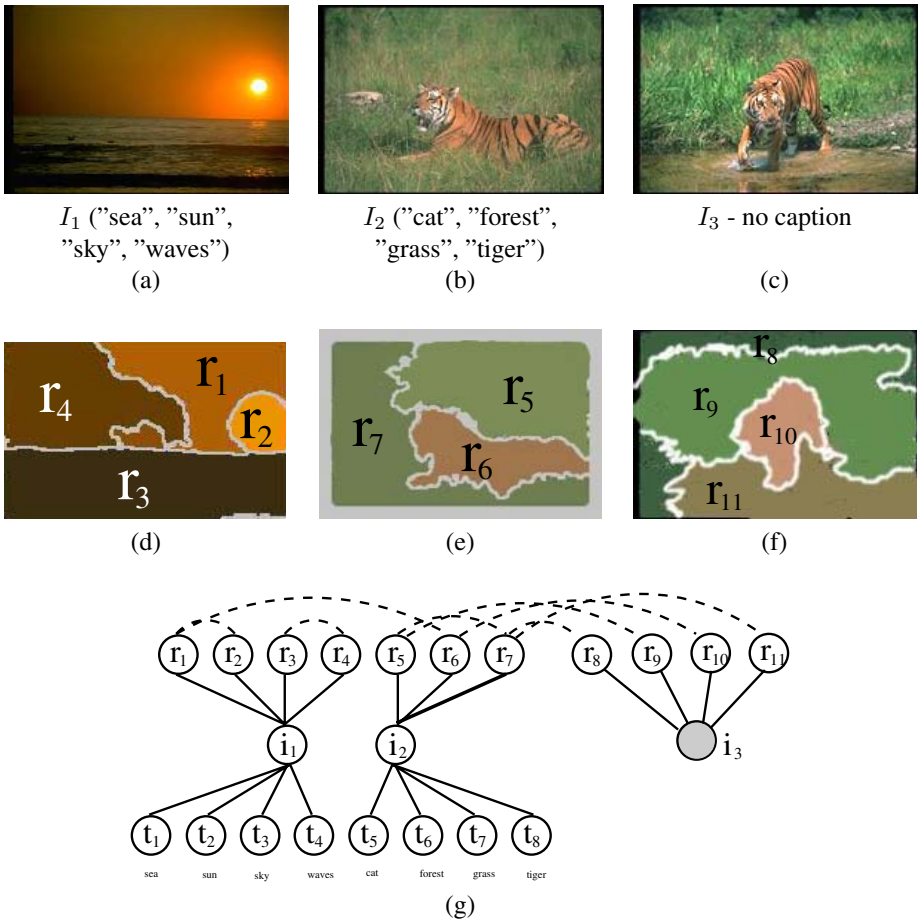


Figure 1: Three sample images, two of them annotated; their regions (d,e,f); and their GCap graph (g). (Figures look best in color.)

We need to decide on a threshold for the "closeness". There are many ways, but we decided to make the threshold adaptive: for each feature-vector, choose its k nearest neighbors, and associate them by connecting them with edges. Therefore, the edges added to relate similar regions are called the "nearest-neighbor" links (NN-links). We discuss the choice of k later, as well as the sensitivity of our results to k.

In summary, we have two types of links in our GCap graph: the *NN*-links, between the nodes of two similar regions; and the (*IAV*-links), between an image node and an attribute value (caption term or region feature vector) node.

Figure 1 illustrates our approach with an example:

Example 1 *Consider the captioned image set $\mathcal{I}$=$\{I_1, I_2\}$ and the un-captioned, query image $I_q = I_3$ (Figure 1). The graph corresponds to this data set has three types of nodes: one for the image objects i_j's ($j = 1, 2, 3$); one for the regions r_j's ($j = 1, \ldots, 11$), and one for the terms $\{t_1, \ldots, t_8\}$=$\{$sea, sun, sky, waves, cat, forest, grass, tiger$\}$. Figure 1(g) shows the resulting GCap graph. Solid arcs indicate IAV (Image-Attribute-Value) relationships; dashed arcs indicate nearest-neighbor (NN) relationships.*

In Example 1, we consider only k=1 nearest neighbor, to avoid cluttering the diagram.

We note that the nearest-neighbor relation is not symmetric. This effect is demonstrated in Figure 1, where node r_2's nearest neighbor is r_1 whose nearest neighbor is r_6. Instead of making NN-links as directed links, we retain the NN-links as undirected. The average degree of each region node is $2k$, where k is the number of nearest neighbors considered per region node. This make node r_1 in Figure 1 have a degree of $2k = 2$. In our experiment, each data set has about $50,000$ regions. For $k = 3$, the region node has the average degree 6 and the standard deviation around

2.25.

To solve the auto-captioning problem (Problem 1), we need to develop a method to find good caption words for image $I_q = I_3$. This means that we need to estimate the affinity of each term (nodes t_1, ..., t_8), to node i_3. We discuss the method we proposed next.

Captioning by random walk We propose to turn the image captioning problem into a graph problem. Thus, we can tap the sizable literature of graph algorithms and use off-the-shelf methods for determining how relevant a term node "v" is, with respect to the node of the uncaptioned image "s". Take Figure 1 for example, we want to rank how relevant the term "tiger" (v=t_8) is to the uncaptioned image node s=i_3. The plan is to caption the new image with the most "relevant" term nodes.

We have many choices: electricity based approaches [8, 20]; random walks (PageRank, topic-sensitive PageRank) [6, 11]; hubs and authorities [13]; elastic springs [16]. In this work, we propose to use *random walk with restarts* ("RWR") for estimating the affinity of node "v" with respect to the restart node "s". But, again, the specific choice of method is orthogonal to our framework.

The choice of "RWR" is due to its simplicity and ability to bias toward the restart node. The percentage of time the "RWR" walk spends on a term-node is proportional to the "closeness" of the term-node to the restart node. For image captioning, we want to rank the terms with respect to the query image. By setting the restart node as the query image node, "RWR" is able to rank the terms according to with respect to the query image node.

On the other hand, methods such as "PageRank with a dumping factor" may not be appropriate for our task, since the ranking it produces does not bias toward any particular node.

The "random walk with restarts" (RWR) operates as follows: to compute the affinity of node "v" to node "s", consider a random walker that starts from node "s". At every time-tick, the walker chooses randomly among the available edges, with one modification: before he makes a choice, he goes back to node "s" with probability c. Let $u_s(v)$ denote the steady state probability that our random walker will find himself at node "v". Then, $u_s(v)$ is what we want, the affinity of "v" with respect to "s".

Definition 1 *The affinity of node v with respect to starting node s is the steady state probability $u_s(v)$ of a random walk with restarts, as defined above.*

For example, to solve the auto-captioning problem for image I_3 of Figure 1. We can estimate the steady-state probabilities $u_{i_3}(v)$ for all nodes v of graph GCap. We can keep only the nodes that correspond to terms, and report the top few (say, 5) terms with the highest steady-state probability as caption words. The intuition is that the steady-state probability is related to the "closeness" between two nodes: in Figure 1, if the random walker with restarts (from i_3) has high chance of finding himself at node v, then node v is likely to be the correct caption for the query image I_3.

2.2 Algorithms

In this section, we summarize the proposed GCap method for image captioning. GCap contains two phases: the graph-building phase and the captioning phase.

Input: a set of captioned images $\mathcal{I}$=$\{I_1, \ldots, I_n\}$ and an uncaptioned image I_q.
Output: the GCap graph for $\mathcal{I}$ and I_q.
1. Let $\mathcal{R}$=$\{r_1, \ldots, r_{N_R}\}$ be the distinct regions appeared in $\mathcal{I}$. Let $\{t_1, \ldots, t_{N_T}\}$ be the distinct terms appeared in $\mathcal{I}$. 2. Similarly, Let $\{r'_1, \ldots, r'_{N_R(I_q)}\}$ be the distinct regions in I_q.
3. Create one node for each region r_i's, images I_i's, and terms t_i's. Also, create nodes for the query image I_q and its regions r'_i's. Totally, we have N=N_I+N_R+N_T+1+$N_R(I_q)$ nodes.
4. Add NN-links between region nodes $V(r_i)$'s, considering only the k nearest neighbors.
5. Connect each query region node $V(r'_i)$ to its k "nearest" training regions $V(r_j)$'s.
6. Add IAV-links between image nodes $V(I_i)$'s and their region/term nodes, as well as between I_q and r_i's.

Figure 2: Algorithm-G: Build GCap graph

The overview of the algorithm is as follows. First, build the GCap graph using the set of captioned images $\mathcal{I}$ and the query image I_q (details are in Figure 2). Then, for each caption word w, estimate its steady-state probability $u_{V(I_q)}(V(w))$ for the "random walk with restarts", as defined above. Recall that $V(w)$ is the node that corresponds to term w.

The computation of the steady-state probability is very interesting and important. We use matrix notation, for compactness. We want to find the most related terms to the query image I_q. We do an RWR from node $V(I_q)$, and compute the steady state probability vector $\overrightarrow{\mathbf{u_q}}$= $(u_q(1), \ldots, u_q(N))$, where N is the number of nodes in the GCap graph.

The estimation of vector $\overrightarrow{\mathbf{u_q}}$ can be implemented efficiently by matrix multiplication. Let $\mathbf{A}$ be the adjacency matrix of the GCap graph, and let it be column-normalized. Let $\overrightarrow{\mathbf{v_q}}$ be a column vector with all its N elements zero, except for the entry that corresponds to node $V(I_q)$; set this entry to 1. We call $\overrightarrow{\mathbf{v_q}}$ the "restart vector". Now we can formalize the definition of the "affinity" of a node with respect

to the query node $V(I_q)$ (Definition 1).

Definition 2 *(Steady-state vector) Let c be the probability of restarting the random walk from node $V(I_q)$. Then, the N-by-1 steady state probability vector, $\overrightarrow{\mathbf{u_q}}$, (or simply, steady-state vector) satisfies the equation:*

$$\overrightarrow{\mathbf{u}_q} = (1-c)\mathbf{A}\overrightarrow{\mathbf{u}_q} + c\overrightarrow{\mathbf{v}_q}. \quad (1)$$

We can easily show that

$$\overrightarrow{\mathbf{u}_q} = c\left(\mathbf{I} - (1-c)\mathbf{A}\right)^{-1}\overrightarrow{\mathbf{v}_q}, \quad (2)$$

where $\mathbf{I}$ is the $N \times N$ identity matrix. The pseudo code of captioning an uncaptioned image I_q is shown in Figure 3. We note that for image captioning, we consider one test image at a time. That is, the GCap graph will always have only one uncaptioned image node (gray-node in Figure 1). The graphs for different test/query images have the same "core" part constituted by the captioned images, but differ at the part where the query image node is connected to the core. We note that building the "core" part of the graph can be done efficiently. Besides, after the "core" part is ready, adding the part relating to a specific query image takes relatively "no" time in practice.

The "core-and-addition" structure of the GCap graph provides opportunities for generality. For example, GCap can be easily extended to caption groups of images like, e.g., a set of video frames, as we discuss later.

Input: a GCap graph of a captioned image set $\mathcal{I}$ and a query image I_q.
Output: the best p caption terms
1. Let $\overrightarrow{\mathbf{v_q}}$=0, for all its N entries, except a "1" for the entry of $V(I_q)$, the node for the query image I_q.
2. Let $\mathbf{A}$ be the adjacency matrix of the GCap graph. Normalize the columns of $\mathbf{A}$ (i.e., make each column sum to 1).
3. Initialize $\overrightarrow{\mathbf{u_q}}$=$\overrightarrow{\mathbf{v_q}}$.
4. while($\overrightarrow{\mathbf{u_q}}$ has not converged)
4.1 $\overrightarrow{\mathbf{u_q}}$ = (1-c)$\mathbf{A}\overrightarrow{\mathbf{u_q}}$ + c$\overrightarrow{\mathbf{v_q}}$
5. Caption I_q with the p terms t_i's which have the highest $u_{V(I_q)}(V(t_i))$ value (t_i's *affinity* to I_q).

Figure 3: Algorithm-IC: Image captioning

2.3 Scalability

Let N_R be the number of regions extracted from all the captioned images $\mathcal{I}$, E be the number of edges in the GCap graph, $N_R(I_q)$ be the number of regions in the query image I_q. and $costNN(N_R)$ be the cost of performing a nearest-neighbor search in a collection of N_R feature vectors.

Lemma 1 *The total training time, T_{train}, of GCap is linear to number of edges E and super-linear on the number of regions N_R:*

$$T_{train} = N_R * costNN(N_R) + E * O(1) \quad (3)$$

Proof: At the *training phase*, Algorithm-G is used to construct the GCap graph. We count only the cost of building "core" of the GCap graph here, the cost of the "addition" part is considered in the testing phase (Lemma 2). To determine the nearest-neighbor links (NN-links), we perform a k nearest-neighbor (k-NN) search on each region. These searches (of cost "$costNN(N_R)$") can be accelerated using an index structure, like an R+-tree [22]. **QED**

Lemma 2 *The overall cost of GCap for captioning a test image , T_{test}, is linear on the number of edges:*

$$T_{add} = N_R(I_q) * costNN(N_R) \quad (4)$$
$$T_{test} = T_{add} + maxIter * O(E) \quad (5)$$
$$= O(E) \quad (6)$$

Proof: In the testing (i.e., captioning) phase, we build the addition to the "core" of the GCap graph, and estimate the steady state probability vector $\overrightarrow{\mathbf{u_q}}$ for a test image I_q. Addition to the GCap graph takes only $N_R(I_q)$ nearest-neighbor searches (usually $N_R(I_q) < 10$), and the total time $N_R(I_q) * costNN(N_R)$ is negligible to the estimation of $\overrightarrow{\mathbf{u_q}}$. $\overrightarrow{\mathbf{u_q}}$ is estimated iteratively, until the estimate stabilizes. The estimate is considered stabilized if the L_1-norm between consecutive round is below some small threshold (e.g., 10^{-9}). In our experiments, the number of iterations to converge ($maxIter$) is typically small (e.g., less than 20), or it can be set to have a upper bound (e.g., 100). In other words, $maxIter$ is of order O(1). For each iteration, a sparse matrix multiplication is performed and costs $2 * E = O(E)$ operations (exactly the number of nonzero elements in the sparse $\mathbf{A}$). **QED**

Although fast already (linear on the database size), the proposed algorithm "Algorithm-IC" can be even further accelerated. We can tap the old and recent literature of fast solutions to linear systems, to achieve fast approximations. We can do the matrix inversion and solve the equation 2; or we can use a low-rank approximation [1, 21]. For matrices that have block-diagonal shapes, we can use the methods by [24]. Given that this area is still under research, we only point out that our approach is modular, and it can trivially include whichever is the best module to do the fast matrix inversion.

2.4 Generality

As mentioned in the introduction, GCap is a general framework, and can handle many more tasks than auto-captioning. We elaborate on three of the possible ways to generalize it:

1. *Other correlations:* Within the auto-captioning problem, GCap can estimate the strength of correlation between *any* two pair of nodes in the graph. Currently we force the first node to be of type "image", while the second node is of type "term". Nothing stops GCap from estimating "term"-"term" correlations or "term"-"image" correlations (e.g., given a term "tiger", what is the most representative image), as well as "term"-"region" correlations (e.g., given a term "tiger", what is the most representative region).

2. *Group captioning:* Within the auto-captioning problem, GCap can find good caption words for a *group* of non-captioned images, say, I_{q1}, I_{q2}, ... The idea is to extend the RWR so that, when it restarts, it randomly restarts from one of the nodes of I_{q1}, I_{q2}, ..., with equally probability.

3. *Arbitrary multimedia setting:* Our GCap method can handle any set of multimedia objects. For example, suppose we have a collection of video clips, each with (a) audio track (b) text (script), and, of course, (c) a succession of frames. Suppose that we want to find the typical sound-track that corresponds to bright images (probably, commercials). Suppose that we are provided one similarity function upon audio segments and one on video frames, by domain experts. In this setting, GCap can build a graph with 4 types of nodes, one for the video clips, one for audio features (e.g., wavelet coefficients), one for script words, and one for video features. Also, the NN-links and IAV-links are well-defined by the given data set and the similarity functions.

3. Experimental Results

In this section, we show experimental results to address the following questions:

- Quality: How does the proposed GCap method perform on captioning test images?
- Parameter defaults: How to choose good default values for the k and c parameters?
- Generality: How well does GCap capture other cross-media correlations? For example, how well does it capture the same-media correlations (say, term-term, or region-region correlations)? Furthermore, how well does the "group captioning" (subsection 2.4) perform?

In our experiment, we use 10 image data sets from Corel, which are commonly used in previous work [9]. In average, each data set has around 50,000 regions, 5,200 images, and 165 words in the captioning vocabulary. The resulting GCap graph has around 55,500 nodes and 180,000 edges. There are around 1,750 query (uncaptioned) images per data set.

3.1 Quality

For each test image, we compute the captioning accuracy as the percentage of caption terms which are correctly predicted. For a test image which has p correct caption terms, GCap will predict also p terms. If l terms are correctly predicted, then the captioning accuracy for this test image is defined as $\frac{l}{p}$.

Figure 4(a) shows the captioning accuracy for the 10 data sets. We compare our results (white bars) with the results reported in [9] (black bars). The method in [9] models the image captioning problem as a statistical translation problem and solves it with an probabilistic model using expectation-maximization (EM). We refer to their method as the "EM" approach. In average, GCap (with c=0.66, k=3) achieves captioning accuracy improvement of 12.8 percentage points, which corresponds to a relative improvement of 58%.

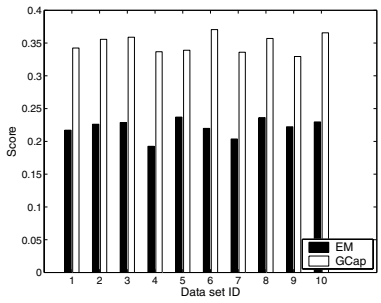
We also compare the captioning accuracy with even more recent machine vision methods [3]: the Hierarchical Aspect Models method ("HAM"), and the Latent Dirichlet Allocation model ("LDA"). The reported results of HAM and LDA are based on the same data set as we used here. Figure 4(b) compares the *best* average captioning accuracy among the 10 data sets reported by the HAM and LDA [3], along with that of GCap (with c=0.66, k=3). Although both HAM and LDA improve on the EM method, they both lose to our generic GCap approach (35% accuracy, versus 29% and 25%). It is also interesting that GCap also gives significantly lower variance, by roughly an order of magnitude: 0.0002 versus 0.002 and 0.003.

Figure 5 shows some examples of the captions given by GCap. For the example query image I_3 in Figure 1, GCap captions it correctly (Figure 5(a)). Note that the GCap graph used for this experiment is not the one shown in Figure 1, which is for illustration only. In Figure 5, GCap surprisingly gets the word "mane" correctly (b); however, it mixes up hand-made objects ("buildings") with "tree" (c).

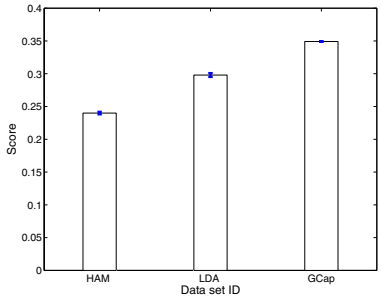
3.2 Parameter defaults

We experiment to find out how would different values of the parameters c and k affect the captioning accuracy. In short, GCap is fairly insensitive to both parameters.

Figure 6(a) shows the captioning accuracy of GCap using different values of restart probability c. The parameter k is fixed at 3. The accuracy reaches a plateau as c grows from 0.5 to 0.9, which indicates the proposed GCap method



(a) scores for EM and GCap



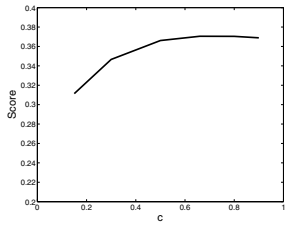
(b) Scores for HAM, LDA, GCap

Figure 4: Comparing GCap with EM, HAM and LDA. In all cases, GCap used $c = 0.66$ and $k = 3$. Results of EM, HAM and LDA are those reported with the best settings [9, 3]. (a) score of EM in dark, against GCap in white, for the 10 Corel datasets (b) scores for HAM (left) and LDA (center): accuracy (mean and variance, over the 10 data sets). LDA:(0.24,0.002); HAM:(0.298,0.003); GCap:(0.3491, 0.0002).

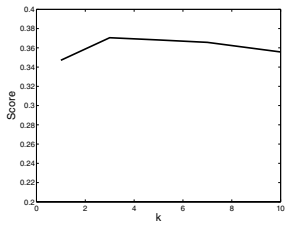
Query			
Truth	cat, grass, tiger, water	mane, cat, lion, grass	sun, water, tree, sky
GCap	grass, cat, tiger, water	lion, grass, cat, mane	tree, water, buildings, sky
	(a)	(b)	(c)

Figure 5: Sample captions generated by GCap . The predicted caption terms by GCap are sorted by the estimated affinity values to the query image. (Figures look best in color.)

is insensitive to the choice of c. We show only the result on one data set "006", the results on other data sets are similar.



(a) Score vs c



(b) Score vs k

Figure 6: (a) Varying decay factor c (fixed k=3). (b) Varying the number of nearest neighbors k (fixed $c = 0.66$). Data set is "006".

Figure 6(b) shows the captioning accuracy of GCap on data set "006" using different number of nearest neighbors k. The restart probability c is fixed at 0.66. Again, the proposed GCap method is insensitive to the choice of k, where the captioning accuracy reaches a plateau as k varies from 3 to 10. Other data sets also have similar results. Another set of experiments, where c is fixed at 0.9 with k varies, show a graph with plateau at the similar accuracy level as Figure 6(b).

3.3 Generality

GCap works on objects of any type. We design an experiment of finding similar caption terms using GCap. Here we use the "core" part of the GCap graph constructed for automatic image captioning, since there is no query image in this case. To find similar terms to a (query) caption term t, we perform "RWR" with the restart vector having all elements zero, except a "1" for the node $V(t)$. Table 2 shows the similar terms found for some of the caption terms. In the table, each row shows the query caption term at the first column, followed by the top 5 similar terms found by GCap (sorted by the steady-state probability).

Notice that the retrieved terms make a lot of sense. For example, the string "branch" in the caption is strongly related to the forest- and bird- related concepts ("birds", "owl", "night"), and so on. Notice again that we did nothing special: no tf/idf, no normalization, no other domain-specific analysis - we just treated these terms as nodes in our GCap graph, like everything else.

4. Discussion

We are shooting for a method that requires no parameters. Thus, here we discuss how to choose defaults for both our parameters, the number of neighbors k, and the restart probability c.

Term	1	2	3	4	5
branch	birds	night	owl	nest	hawk
bridge	water	arch	sky	stone	boats
cactus	saguaro	desert	sky	grass	sunset
car	tracks	street	buildings	turn	prototype
f-16	plane	jet	sky	runway	water
market	people	street	food	closeup	buildings

Table 2: Semantically similar terms for selected caption terms

Number of Neighbors k In hindsight, the results of Figure 6 make sense: with only k=1 neighbor per region, the collection of regions is barely connected, missing important connections and thus leading to poor captioning performance. On the other extreme, with a high value of k, every region feature vector is directly connected to every other one; the region nodes form almost a clique, which does not distinguish clearly between really close neighbors with those which are just neighbors.

For a medium number of neighbors k, our NN-links apparently capture the neighbors which are really close. Small deviations from that value, make little difference, probably because the extra neighbors we add are at least as good as the previous ones. We suggest that the caption accuracy is not sensitive to k, for a reasonable medium value of k.

Restart probability c For web graphs, the recommended value for c is typically c=0.15 [24]. Surprisingly, our experiments show that this choice does not give good captioning performance. Instead, good quality is achieved for c=0.66. Why is this discrepancy?

We conjecture what determines a good value for the restart probability is the diameter of the graph. Ideally, we want our "random walker" to have a non-trivial chance to reach the outskirts of the whole graph. Thus, if the diameter of the graph is d, the probability that he will reach a point on the "periphery" is probably proportional to $(1-c)^d$.

For the web graph, the diameter is approximately d=19 [2] which implies that the probability $p_{periphery}$ for the random walker to reach a node in the periphery is roughly (let c=0.15)

$$p_{periphery} = (1-c)^{19} = 0.045$$

In the case of auto-captioning, with a three-layer graph, the diameter is roughly d=3. If we demand the same $p_{periphery}$ for our case, then we have

$$\begin{aligned} (1-0.15)^{19} &= (1-c)^3 \\ \Rightarrow c &\approx 0.65 \end{aligned}$$

which is much closer to our empirical observations. Of course, the problem requires more careful analysis - but we are the first to show that c=0.15 is not always optimal for random walks with restarts.

Updating training image sets As more captioned images become available, they can be easily appended to the existing training set. Each new image is represented as an image node with a set of region nodes. The incorporation of a newly available captioned image is simply adding the new image node, the new region nodes and possible new caption term nodes to the existing GCap graph, by the NN-links and IAV-links . Adding the NN-links involves nearest-neighbor searches at each newly added regions, which can be done efficiently with the help of an index structure like R+-tree. Adding the IAV-links is straight-forward: simply connect each newly added image nodes to the term and region nodes it contains. To sum up, the updating of the training set can be done efficiently and incrementally.

Group captioning The proposed GCap method can be easily extended to caption a group of images, where the content of these images are considered simultaneously. One possible application is to caption video-shots, where a shot is represented by a set of keyframes sharing the same story content. Since these keyframes are related, captioning them as a whole can take into account the correlation they share, which is missed when they are captioned separately. Figure 7 shows the results of GCap when applied for "group-captioning" a set of three images. Notice that it found very reasonable terms, "sky", "water", "tree", and "sun".

Images			
Truth	sun, water, tree, sky	sun, clouds, sky, horizon	sun, water
GCap	tree, people, sky, water	water, tree, people, sky	sky, sun
Group	sky, water, tree, sun		

Figure 7: (Group captioning) Captioning terms are sorted by the steady state probability computed by GCap. (Figures look best in color.)

5. Conclusions

We proposed GCap, a graph-based method for automatic image captioning. The method has the following desirable characteristics:

- It provides excellent results and outperforms recent, successful auto-captioning methods (EM, HAM, LDA) (Figure 4).
- It requires no user-defined parameters, nor any other tuning (in contrast to linear/polynomial/kernel SVMs, k-means clustering, etc.). We give good default values for its only two parameters, k and c. We also show empirically that the performance is fairly insensitive to them, anyway.
- It is fast and scales up well with the database size. It can be made even faster, with clever, off-the-shelf matrix algebra methods (equation 2).

Future work could focus on weighting the edges to improve captioning accuracy. Edge weights could take into account the difference between NN-links and IAV-links, as well as the difference of the individual edges. Besides, it will be interesting to apply GCap for more general settings, to discover cross-modal correlations in mixed media databases, as we described earlier in subsection 2.4.

References

[1] D. Achlioptas and F. McSherry. Fast computation of low-rank approximations. In *STOC 01*, pages 611–618, 2001.

[2] A. Albert, H. Jeong, and A.-L. Barabasi. Diameter of the world wide web. *Nature*, 401:130–131, 1999.

[3] K. Barnard, P. Duygulu, N. de Freitas, D. A. Forsyth, D. Blei, and M. Jordan. Matching words and pictures. *Journal of Machine Learning Research*, 3:1107–1135, 2003.

[4] K. Barnard and D. A. Forsyth. Learning the semantics of words and pictures. In *Int. Conf. on Computer Vision*, pages 408–15, 2001.

[5] D. Blei and M. I. Jordan. Modeling annotated data. In *26th Annual International ACM SIGIR Conference*, July 28-August 1, 2003, Toronto, Canada.

[6] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International World Wide Web Conference*, 1998.

[7] P. Carbonetto, N. de Freitas, and K. Barnard. A statistical model for general contextual object recognition. In *Proceedings of ECCV*, 2004.

[8] P. G. Doyle and J. L. Snell. *Random Walks and Electric Networks*. Kluwer.

[9] P. Duygulu, K. Barnard, N. Freitas, and D. A. Forsyth. Object recognition as machine translation: learning a lexicon for a fixed image vocabulary. In *Seventh European Conference on Computer Vision (ECCV)*, volume 4, pages 97–112, 2002.

[10] J. Edwards, R. White, and D. Forsyth. Words and pictures in the news. In *HLT-NAACL03 Workshop on Learning Word Meaning from Non-Linguistic Data*, 2003.

[11] T. H. Haveliwala. Topic-sensitive pagerank. In *WWW2002*, May 7-11 2002.

[12] J. Jeon, V. Lavrenko, and R. Manmatha. Automatic image annotation and retrieval using cross-media relevance models. In *26th Annual International ACM SIGIR Conference*, July 28-August 1, 2003, Toronto, Canada.

[13] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[14] V. Lavrenko, R. Manmatha, and J. Jeon. A model for learning the semantics of pictures. In *NIPS*, 2003.

[15] J. Li and J. Z. Wang. Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(10):14, 2003.

[16] L. Lovasz. Random walks on graphs: A survey. *Combinatorics, Paul Erdos is Eighty*, 2:353–398, 1996.

[17] O. Maron and A. L. Ratan. Multiple-instance learning for natural scene classification. In *The Fifteenth International Conference on Machine Learning*, 1998.

[18] F. Monay and D. Gatica-Perez. On image auto-annotation with latent space models. In *Proc. ACM Int. Conf. on Multimedia (ACM MM)*, Berkeley, November 2003.

[19] Y. Mori, H. Takahashi, and R. Oka. Image-to-word transformation based on dividing and vector quantizing images with words. In *First International Workshop on Multimedia Intelligent Storage and Retrieval Management*, 1999.

[20] C. R. Palmer and C. Faloutsos. Electricity based external similarity of categorical attributes. In *PAKDD 2003*, May 2003.

[21] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent semantic indexing: A probabilistic analysis. In *PODS 98*, 1998.

[22] T. Sellis, N. Roussopoulos, and C. Faloutsos. The R+-tree: A dynamic index for multi-dimensional objects. In *12th International Conf. on VLDB*, pages 507–518, Sept. 1987.

[23] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[24] S. K. Taher Haveliwala and G. Jeh. An analytical comparison of approaches to personalizing pagerank. Technical report, Stanford University, 2003.

[25] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *Proceedings of the ACM CHI*, 2004.

[26] L. Wenyin, S. Dumais, Y. Sun, H. Zhang, M. Czerwinski, and B. Field. Semi-automatic image annotation. In *INTERACT2001, 8th IFIP TC.13 Conference on Human-Computer Interacti on*, Tokyo, Japan July 9-13, 2001.