Querying Users as Oracles in Tag Engines for Personalized Image Tagging

Amandianeze O. Nwana
Cornell University

Tsuhan Chen
Nanyang Technological University Singapore

A learning-to-rank (L2R) formulation of personalized image tagging that leverages the user’s past tag orders as implicit feedback in learning, in a semi-supervised fashion, their tag preferences, which is used in predicting the tag sequence on a new image for the user.

The dominant notion of tag usefulness is that tags should be useful as search query terms for the images they accompany. Therefore, tagging systems generally consider only tags with an unambiguous visual interpretation as useful. As a result, much of the work on image tagging treats it essentially as a multiclass labeling problem, where the classes are typically restricted to objects that are salient and visually recognizable in the image.1,2 Another trend focuses on tag occurrence and co-occurrence statistics, focusing on either tag frequency or tag distinctiveness.3 However, even research following this trend operates under the same notion of tag usefulness, propagating for the most part only tags that have precise and unambiguous visual interpretations. (See the sidebar for further discussion of related work.)

We challenge this dominant notion of tag usefulness, especially under the premise of personalized image tagging. Figure 1 shows an example of an image whose tags don’t fit this dominant notion. When designing user-centric tagging and indexing systems, we need to modify the current notion of tag usefulness (which hinges on their appropriateness as web search query terms) to notions that capture user preferences and behavior. To do this, we must learn what these user preferences are. It is our assumption, as we demonstrate elsewhere,4 that these preferences can be inferred from the tag lists users provided on other images in two ways:

1. the tags that appear on the list are preferred to those that do not, and
2. the order in which the tags are listed implies a preference on the listed tags such that those listed earlier are preferred to those listed later.

Our earlier findings,4 along with a Flickr design change (see www.flickr.com/help/forum/en-us/72157645219834187) and the subsequent reversal of the change based on user feedback about intentional tag ordering, support the second assumption, leading us to believe that user tag preferences can be exploited via tag order.

We propose a new method5 of image tag prediction using a learning-to-rank framework.6 Our method attempts to learn the tag-ranking functions, which we assume are unique and inherent to each user, so that given a set of good candidate tags for an image, we can rank them in a manner that best mimics what the user would have done if limited to that vocabulary. In addition, we propose a semisupervised framework to generate more seed tags, because user-generated tags are typically sparse (at least in comparison to the vocabulary size), which makes it hard for algorithms such as RankSVM6 to learn suitable ranking functions. This new prediction paradigm treats the order in which users have presented tags in the past as clickthrough data, which in the search engine domain serves as useful implicit user feedback. In this sense, the user-ordered tag lists are treated as (noisy and incomplete) oracles.

Problem Formulation
We model the problem of personalized automatic image tagging as a search/retrieval-ranking problem, and employ RankSVM6 to learn our user models (that is, ranking functions).
Most work in image tagging exploits “tag-tag” co-occurrence or correlations to make predictions based either on an initial seed of tags or on “tag-image content” correlations and statistics. Some work has tried to embed both visual features and tags into a shared multi-dimensional space to capture the semantic similarity between tags and images via their distance from one another in the space. These efforts are agnostic to user preferences among tags, but rather emphasize the image’s “preferences” among tags or, better yet, a tag’s appropriateness to a given image.

Merrielle Spain and Pietro Perona define an object’s importance as the probability that it is mentioned first in the tag list for the image, given multiple instances of the tag lists from different users assuming the tag lists are independent and identically distributed. Alexander Berg and his colleagues include the notion of object ontologies, with a focus on attribute detection and scene understanding, to determine the importance of objects and appropriateness of tags in images. But they too focus on global notions of importance, not being able to capture personal preferences of individual users under their framework.

Steffen Rendle and Lars Schmidt-Thieme represent the tag recommendation problem under the framework of tensor factorization. The constraints on their learning objective are, for a given <user, item/image> pair in the training data, the tags mentioned for that pair must be preferred to those that were not. This is similar to our first assumption, but Rendle and Schmidt-Thieme do not try to learn or enforce a relation between tags that appear together, thereby ignoring the structure inherent to the user-provided tag lists—another of our important assumptions.

Marek Lipczak and Evangelos Milios also treat tags as essentially structureless entities (bag of words) and learn, for each user, how to merge tag statistics for each tag from various modalities. Xirong Li and his colleagues, similarly to Lipczak and Milios, propose a method for learning how to weight tag scores from multiple tagging functions to maximize some desired metric (for example, mean average precision) on training data per user. To the best of our knowledge, no other work on personalization, besides ours, treats the user-provided tag lists as anything more than an orderless, structureless set. We believe this is one of the novelties of our work with respect to others on personalization. As mentioned in the main text, our work provides evidence that user tag lists are more than just a bag of words. Our proposed method differs from prior work in that we learn each user’s inherent tagging functions in a semisupervised manner based on previously observed rankings by the user.

There are a few other works on learning-to-rank methods for tagging, but these still treat tag relevance in terms of a tag’s descriptive power and its discriminative power among other descriptive tags. Our method differs from the methods we’ve mentioned in that we are concerned with personalized notions of relevance/preference, and we use the user-provided tag lists as the ground truth of the correct ranking of tags for that user, thereby implicitly “querying” the user.

Further discussion of related work is available elsewhere.

References
Learning to rank using RankSVM. RankSVM learns ranking functions from preference judgments. For simplicity, we define a search session $S$ as the tuple $(q, D_q, C)$, where $q$ is the query, $D_q$ is the set of retrieved documents; and $C \subset D_q$ is the set of clicked documents. From each session $S$ we derive pairwise preference judgments $P_S$ of the form $P_S = \left\{ d_i \succ d_j : d_i \in C, d_j \in D_q \setminus C \right\}$. We use $(d_i, d_j)$ as shorthand for $d_i \succ d_j$ in the rest of the article. Let $S$ be the training set of all observed sessions. Simply stated, the objective of RankSVM is, given $S$, learn a ranking function $\hat{w}$ that minimizes the number of reversed preference judgments over all observed sessions. More concretely,

$$\text{minimize } V(\hat{w}, \xi) = \frac{1}{2} \hat{w} \cdot \hat{w} + C \sum \xi_{i,j,k}$$

subject to

$$\forall (d_i, d_j) \in P_{S_1} : \hat{w} \Phi(q_1, d_i) > \hat{w} \Phi(q_1, d_j) + 1 - \xi_{i,j,1}$$

$$\forall (d_i, d_j) \in P_{S_2} : \hat{w} \Phi(q_2, d_i) > \hat{w} \Phi(q_2, d_j) + 1 - \xi_{i,j,2}$$

$$\forall i \neq j : \xi_{i,j,k} > 0,$$

where $\hat{w}$ is the (linear) ranking function; $\Phi(q, d)$ is a mapping onto features describing the query and document; $\xi_{i,j,k}$ is a slack variable that allows some degree of error in the learned preference judgment between documents $i$ and $j$ for session $k$; and $C$ is the parameter that controls the trade-off between minimizing training error and generalization (that is, reducing overfitting).

Given a new query $q'$ to rank the set of retrieved documents $D_{q'}$, one only has to compute $\hat{w} \Phi(q', d) \forall d \in D_{q'}$ and sort in descending order.

Tag-ranking formulation. We adapt the RankSVM framework to our personalized automatic image-tagging problem in the following way: a session $S$ is represented by tuple $(I, V, T)$, where $I$ is the image being tagged and is analogous to the text query $q$; $V$ is the set of available tags (or vocabulary), analogous to the documents; and $T$ is the set of tags that were given by the user, analogous to the clicked pages.

We make the following assumptions on $V$ and $T$.

The set of tags $T$ has inherent structure in that the tags that are mentioned earlier are implicitly preferred to those mentioned later. Note that this assumption is analogous to saying that the pages that were clicked first are preferred to those clicked later; however, this assumption would likely not be good in that setting, because users typically estimate, based on limited information (title and excerpt), that they will find the page relevant or get some utility from it. In fact, it is likely that they keep clicking because they haven’t found a satisfactory page, and they will stop either when they find a more satisfactory page or reach some upper bound on time spent searching. In our setting, however, the users have potentially full
information—that is, knowledge of both the image content (this makes sense, because they observe the image before tagging) and the vocabulary (again another sensible assumption, especially in light of personalization).

The elements in \( T \) are also implicitly preferred to the elements in \( V \backslash T \). This is similar to our earlier assumption that given a vocabulary of tags (or set of documents), the subset of tags the user mentioned for the image are analogous to the subset of documents the user would have clicked on.

The set \( V \) also has structure, in that we can define functions over \( V \) that measure the relevance of the tags in \( V \) to the image \( I \), so that we can order the set \( V \) for image \( I \).

### Personalization Model and Query Mapping

In this work, we follow an implicit (feature similarity-based) approach to mining tags through visually similar images,\(^2\) in contrast to explicit (classifier and rule-based) approaches. To that end, every image in our dataset is represented by Euclidean distance between image descriptors as extracted in training. This lets us use the standard shorthand \( V \) for \( V(I, u) \) where convenient and unambiguous.

So, given a user session, \( S = (I, V, T) \), where \( T \) is represented as an ordered set, \(<t_1, t_2, \ldots, t_{|T|}>\), under the assumptions presented earlier, \( P_S \) our implied preference judgments become

\[
S(I, u) = \bigcup_{V(I)} O_S(T) \cup O_S(V) \cup O_S(V)
\]

where \( O_S(T), O_S(V), \) and \( O_S(V) \) are the preference judgments derived from our first, second, and third assumptions, respectively:

\[
O_S(T) = \{ (t_i, t_j): t_i, t_j \in T, i < j \} \\
O_S(V, T) = \{ (t_i, t_j): t_i, t_j \in V \backslash T \} \\
O_S(V) = \{ (t_i, t_j): t_i, t_j \in V \backslash T, v_u(t_i, I, m) \}
\]

With this definition of \( P_S \), we can learn the personalized ranking functions in a semisupervised way by augmenting the observed order preferences in the set \( T \) with other relevant unobserved order preferences from \( V \).

To learn a ranking function for the tags, according to Equations 1–3, we need to define a mapping \( \Phi(q, t) \) from the query image and candidate tag to some feature space. To that end, we use the word2vec tool (https://code.google.com/p/word2vec) to learn vector representations for the tags. We denote the tag \( t \)'s vector representation from word2vec as \( w_2v(t) \). Each observed image in our training set represents a document, with the accompanying tags as the

\[
\Phi(q, t) = w_2v(t)
\]
words in the document. We train our word2vec model using the skip-gram architecture, and we embedded the tags in 100-dimensional space.

We also included the following tag statistics as features:

- $mp(t)$: The tag’s mean position on the tag lists it appears on.
- $vp(t)$: The variance of the tag’s position on the tag lists it appears on.
- $cb(t)$: The probability the tag is mentioned on any tag list.

Finally, we have $\Phi(q, t) = w2v(t) :: mp(t) :: vp(t) :: cb(t)$, where :: is the append operator. We do not use the query $q$ (that is, the image) in our mapping function. Rather, we leave it to future work.

**Experiments and Results**

In evaluating our model, we discuss choice of dataset, choice of baseline, and choice of evaluation metric.

**Dataset**

We work with the NUS-WIDE dataset, which is a subset of 269,648 images from Flickr. For each image in the dataset, we know, via the Flickr API, the user that uploaded the image and the sequence of the tags with which the user annotated the image. Because we are particularly concerned with personalization, we only select images from this database where the user who uploaded the image has at least six images in this dataset, similar to other work. This results in approximately 91,400 images from 5,000 users. We split the dataset into training and test partitions by randomly assigning half of each user’s images to the training and half to the test set. For each image, we only retain the tags that occurred frequently enough across the dataset to make some sort of meaningful inference on the tags. We decided to work with tags that occurred at least 50 times in the dataset. This results in a vocabulary of 5,326 unique tags.

**Baselines**

We use two baselines for comparison, both of which are evaluated on the NUS-WIDE dataset.

The first baseline is from Li and his colleagues, which was considered the state of the art in personalized image tagging prior to the second baseline method. Their main idea is that for a given tag, each user has two weighting variables, one to weight how much to rank the tag according to its frequency among the user’s past images independent of visual content, and the other to weight the tag’s uniqueness according to its frequency among visually similar images versus its frequency from all previous images (not just the user’s). These weights are user dependent, and together they determine a tag’s score. Li and his colleagues’ main contribution is a method to optimize these weights.

We implement their method using the two tagging functions corresponding to the two factors described earlier and find weights for the PersonalPreference and Visual factors. For the visual factor, we also use the 500-D bag of words based on SIFT descriptors for consistency. Li and his colleagues ignore the inherent structure of the tag lists and treat a tag list essentially as a bag of words. We denote this method $XE$ (for cross-entropy).

The second baseline is a heuristic method, which demonstrates another use of the pairwise tag biases observed from users’ past tagging histories to rerank existing tag functions. The main idea is that if the pairwise order $<A, B>$ is observed significantly more often than $<B, A>$ for a user $u$, then strictly enforce $<A, B>$ in future predictions for that user while preserving pairwise relationships among tag pairs that need no reordering (because either there is no strong preference or they are already in the correct order). This method takes the tags from some default rank $D$ and reranks them according to the strength of pairwise preferences from the training set. We define the pairwise strength as:

$$p_{ab} = \frac{\text{# times tag } a \text{ is mentioned before tag } b}{\text{# times tags } a \text{ and } b \text{ are mentioned together}}$$

For the purposes of this experiment, we only considered preference strengths greater than 0.8 (chosen empirically) to prevent overfitting. We can represent these strengths among candidate tags succinctly in a directed constraint graph, and obtain a ranking that respects these constraints through a topological sort (using the default ranking $D$ to resolve ties and cycles). One drawback of this heuristic method is that it is not generalizable to tag pairs that never occurred together in data, whereas the method
we propose here. We denote this heuristic baseline as *PT_Rerank*.

**Metrics**

Because we assume that the order in which a user tags an image is of some importance, we would like a metric that takes order into account. Since we are interested in personalization, we treat the user-provided tag list order as the ground truth order. More concretely, if we have an ordered set of tags, \( \{t_1, \ldots, t_k\} \), we define the importance or relevance of each tag as follows: Let \( \text{rank}^k(t_i) \) be the rank of \( t_i \) with respect to the ground truth order and only if the tag is not among the ground truth:

\[
\text{rel}(t_i) = \frac{1}{\text{rank}^k(t_i)} \quad \forall t_i,
\]

Equation 7 is defined as the reciprocal rank (or zipf rank). In simple terms, it states that the \( i \)th tag presented by the user (that is, ground truth) is only \( 1/i \) times as relevant as the first presented tag.

More common metrics, such as precision, recall, and average precision, assume that all tags are equally relevant, so we do not utilize those metrics here. Instead, we use the more appropriate discounted cumulative gain (DCG). This is a common metric used in evaluating search engine results. For an ordered set, \( T = \{t_1, \ldots, t_k\} \), such that \( i < j \rightarrow t_i \succ t_j \), we define the DCG with respect to the ground truth as

\[
\text{DCG}(T) = \text{rel}(t_1) + \sum_{i=2}^k \frac{\text{rel}(t_i)}{\log_2(i)}.
\]

Equation 8 is defined as the DCG of \( T \) with respect to the ground truth. To compute the DCG, we provide a threshold \( \frac{1}{k} \) to include the top \( k \) tags. Notice that in Equation 8, \( \text{rank}^k(t_i) = i \) if and only if the ranked list \( T \) is exactly the same as the ground truth. This metric is called “discounted” because the later we include a tag in our ranking, the less gain we get from it (that is, its relevance is discounted by the inverse of the log of its position in the ranking, not the ground truth).

Like the precision, recall, and average precision metrics, we can also parameterize the DCG metric to calculate the \( \text{DCG}@k \)—that is, calculate the metric using only the first \( k \) entries of the ranked lists. Let \( T[:k] \) be the first \( k \) entries of \( T \), then, \( \text{DCG}@k(T) = \text{DCG}(T[:k]) \).

**Experiment Setup**

In our experiments’ training phase, for every user \( u \) we produced a ranking example from each image \( i \) that the user tagged in the training set. We took the supervised tag order for that image \( T_i \) as the user-provided tag list with order preserved. We then produced the semisupervised set \( \hat{V}_i \) from the tags mined from the image’s most visually similar neighbors, according to Equation 4. Next, we learn the user’s learned ranking function \( \hat{w}_u \) by solving the RankSVM objective using the constraints described previously, with the SVM-Light software package.

During the testing phase, we use the nearest visual neighbors of a query image \( i \) to generate candidate tags that need to be ranked by the user’s learned ranking function \( \hat{w}_u \) from the training phase. These candidate tags are from \( \hat{V}(i, u) \), according to Equation 4.

For our RankSVM formulation, we need to choose the regularization term \( C \), which controls the size of the slack variable and hence the tradeoff between training error and generalization. We tried a few values of \( C \) between 0.01 and 10 but the choice did not seem to impact the results, so we chose \( C = 0.01 \) for our experiments, because it was the fastest for training among the values we tried.

Another design decision was how much data to use in creating the preference orders \( P_3 \) per training image, as described earlier. Given a session \( (I, V, T) \), users tend to give too few tags. However, using the entire set \( V \) as negative tag examples—used in creating \( O_3(T, V) \) in Equation 6—might slow the training phase due to too many pairwise constraints. It might also create noisy/meaningless constraints, because not all the tags in our vocabulary are related to the image in question. To address these issues, we decided to focus not on the whole dictionary but instead on those tags that are related/relevant as denoted by \( V \) in Equation 4. That is, we set \( V \) to \( V \\hat{V} \).

We also analyzed the effect of the number of tags given per example by defining \( \hat{T} = T :: \hat{V} \), which is the tag order defined by Equation 4 (semisupervised data) appended to the ground truth user-generated tags. To study the effect of the number of tags used we parameterize \( T(n) = T[:n] \) as the first \( n \) elements of \( \hat{T} \). With some abuse of notation, this corresponds to setting \( T = T(n) \) and \( V = V = \emptyset \) in Equations 5 and 6. Because both the training and testing phases of our method require nearest-neighbor search, we must also determine the number of nearest neighbors to consider. In this work, we used \( N = 50 \) nearest neighbors.

Another design decision was to group the tags in \( \hat{T} \) into levels of relevance, with tags at
the same relevance level having no preference among one another, but tags at lower levels preferred to tags in higher levels. We assigned the first five tags in $^T$ to levels 1–5, respectively, and then grouped the next set of five tags into increasing levels (for example, tags 6–10 are level 6, and tags 11–15 are level 7). This helps to prevent overfitting due to potential noise in the data, and reduces the number of constraints to enforce in training each user, thereby speeding up training.

**Evaluations**

We evaluate our method using the DCG metric on both a per-image and a per-user basis. Because we are concerned with personalization, we want to know the average performance of our method for each user compared to the baselines. We also present the DCG@10, because in our dataset, the average number of tags provided per user is about 10.

We explore the effect of using only supervised orders, $QUOTE(|T|)$, without supervised orders, $QUOTE(|V|)$, and also the effect of using all the tags generated during the semisupervised step, $QUOTE(\infty)$. We also explore the effect of the number of tags $n$ per image when training our method, hereafter referred to as $QUOTE(n)$.

To make sure that our method truly captures personalization, we also compare it to a modified version, which, in the testing phase, uses another user's ranking function (randomly) to sort the tags mined from the nearest neighbors. We refer to this setting as $rQUOTE$.

**Observations**

As Figures 2 and 3 show, the number of ordered tags (clickthrough data) provided per training example profoundly affects the performance of our method as expected. The more ordered tags provided, the better performance we get up to a
point (100 tags), where we observe a slight degradation and then saturation. This degradation and saturation is likely because as we add more tags to the training images in our semisupervised method, according to Equation 4, we start to include some less relevant tags. As the number of tags increases past 100, our method seems to asymptotically approach similar performance to when all the semisupervised provided tags are included for training. We also notice that at 40 training tags, our method already begins to match and outperform the PT_Rerank baseline, which outperforms the previous state of the art. Given our vocabulary’s size (more than 5,000 tags), we do not think that 40–100 tags per image in training is excessive, especially when generated cheaply without the need for expert knowledge.

The observations for the image-level averages (Figure 2) and the user-level averages (Figure 3) are similar. This shows that we do not just learn global notions of tag preference, but, more importantly, personalized tag preferences. So, for the average user in our dataset, we learn a ranking function that leverages the user’s inherent ranking and preference of tags to improve personalized image tagging. Tables 1 and 2 show the average percentage improvement of our method over the PT_Rerank baseline, and we again observe that using more tags improves our method.

As Table 3 shows, using the constraints without the supervised order performs better than when we only use the supervised constraint. As we stated earlier, this result is not surprising because there are typically far fewer supervised constraints, because users tend to only provide a small number of tags (approximately 10 on average). However, as expected, the combination of supervised and unsupervised constraints leads to an improvement over both. We also see that our method learns personal models, because rQUOTE, which ranks queries from one user with another random user’s ranking function, performs drastically worse.

To verify that the improvements reported in Tables 1–3 are statistically significant, we performed two-sided student t-tests. All our improvements have p-values of less than 0.001, which implies that our findings are indeed statistically significant. Figure 4 provides qualitative examples. As the figure shows, our method often performs better than previous methods, both in better results and in the relative ranking of tags found in the ground truth.

| Table 1. Average DCG@10 percentage improvement over the PT_Rerank baseline.* |
|-----------------|--------|--------|--------|--------|--------|
| QUOTE(k)        | $k = |T|$ | $k = \infty$ | $k = 10$ | $k = 100$ | $k = 200$ |
| Per image       | -36.0  | 5.9    | -38.1  | 11.3    | 7.7     |
| Per user        | -41.1  | 0.003  | -35.7  | 4.3     | -0.001  |

*All improvements were calculated to have p-value < 0.001 using a two-sided student’s t-test.

| Table 2. Average DCG percentage improvement over the PT_Rerank baseline.* |
|-----------------|--------|--------|--------|--------|--------|
| QUOTE(k)        | $k = |T|$ | $k = \infty$ | $k = 10$ | $k = 100$ | $k = 200$ |
| Per image       | -22.1  | 5.0    | -24.9  | 7.7     | 6.1     |
| Per user        | -23.4  | 4.6    | -21.1  | 7.1     | 4.7     |

*All improvements were calculated to have p-value < 0.001 using a two-sided student’s t-test.

| Table 3. Comparison of the different components of our proposed method.* |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| DCG per image   | $QUOTE(T)^{\dagger}$ | $QUOTE(V)^{\ddagger}$ | $QUOTE(\infty)^{\S}$ | $rQUOTE(T)^{\†}$ |
| DCG@10 per image| 0.632            | 0.817            | 0.852            | 0.455           |
| DCG per user    | 0.378            | 0.588            | 0.626            | 0.173           |
| DCG@10 per user | 0.563            | 0.728            | 0.770            | 0.408           |
| DCG@10 per user | 0.348            | 0.544            | 0.593            | 0.168           |

*All improvements were calculated to have p-value < 0.001 using a two-sided student’s t-test.

$QUOTE(T)^{\dagger}$ measures the performance of just the supervised.

$QUOTE(V)^{\ddagger}$ measures the performance of the nonsupervised order.

$QUOTE(\infty)^{\S}$ measures the combined performance.

$rQUOTE(T)^{\†}$ measures the performance when not using user-specific models.

Future Directions

Our method is also more efficient than the previous state of the art. We use $O(#Users)$ parameters, whereas Li and his colleagues require $O(#Users \times #Tags)$, and PT_Rerank requires $O(#Users \times #Tags^2)$ parameters to train the respective models. For Li and his colleagues, the training time per user was 1 minute (not including I/O), whereas for ours, it was under 1 second (with I/O time).

An interesting future direction is to jointly embed the image features and tag features to learn a better mapping that leverages both the image content and tag meaning using deep recurrent and convolutional neural networks. It would also be interesting to explore the
cognitive processes driving a user’s tagging order. Do the users follow an ontology in a top-bottom or bottom-top manner? Is there some latent hierarchy of tags, or maybe parts of speech? What role does sentiment play in tagging? Another interesting direction is the cognitive dimensions that drive tag ordering, and how these cognitive dimensions contribute to users’ tagging choices. Which cognitive dimensions influence personalization more than others?

Another interesting question is whether we could pose the auto-tagging problem as a machine-training instance: Given an image, output the most likely sequence of words (tags)? When the restriction to visually relevant tags is lifted, the personalized image-tagging problem becomes richer, letting us leverage structure and signals in novel ways as we have demonstrated here.

Acknowledgments

This special issue is a collaboration between the 2016 IEEE International Symposium on Multimedia (ISM 2016) and IEEE MultiMedia. This article is a revised version of “QUOTE: ‘Querying’ Users as Oracles in Tag Engines: A Semi-Supervised Learning Approach to Personalized Image Tagging,” presented at ISM 2016.

References


Amandianeze O. Nwana is a software engineer at YouTube. (Research for this article was done while he was at Cornell University.) His research interests include applied machine learning, online social network analysis, and user behavioral modeling, especially with regard to latent or implicit user preferences. Nwana received his PhD in electrical and computer engineering from Cornell University, where he was a member of the Advanced Multimedia Processing Laboratory. He is a member of the Bouchet Honor Society. Contact him at aon3@cornell.edu.

Tsuhan Chen is dean of engineering at Nanyang Technological University Singapore. His research interests include pattern recognition, machine learning, and computer vision. Chen has a PhD in electrical engineering from the California Institute of Technology. He is a fellow of IEEE. Contact him at tsuhan@ntu.edu.sg.

**Experimenting with your hiring process?**

Finding the best computing job or hire shouldn’t be left to chance.

IEEE Computer Society Jobs is your ideal recruitment resource, targeting over 85,000 expert researchers and qualified top-level managers in software engineering, robotics, programming, artificial intelligence, networking and communications, consulting, modeling, data structures, and other computer science-related fields worldwide. Whether you’re looking to hire or be hired, IEEE Computer Society Jobs provides real results by matching hundreds of relevant jobs with this hard-to-reach audience each month, in *Computer* magazine and/or online-only!

http://www.computer.org/jobs

The IEEE Computer Society is a partner in the AIP Career Network, a collection of online job sites for scientists, engineers, and computing professionals. Other partners include *Physics Today*, the American Association of Physicists in Medicine (AAPM), American Association of Physics Teachers (AAPT), American Physical Society (APS), AVS Science and Technology, and the Society of Physics Students (SPS) and Sigma Pi Sigma.