Automatic Image Annotation using Deep Learning Representations

Venkatesh N. Murthy, Subhransu Maji, R. Manmatha
School of Computer Science
University of Massachusetts, Amherst, MA, USA.
{venk, smaji, manmatha}@cs.umass.edu

ABSTRACT

We propose simple and effective models for the image annotation that make use of Convolutional Neural Network (CNN) features extracted from an image and word embedding vectors to represent their associated tags. Our first set of models is based on the Canonical Correlation Analysis (CCA) framework that helps in modeling both views – visual features (CNN feature) and textual features (word embedding vectors) of the data. Results on all three variants of the CCA models, namely linear CCA, kernel CCA and CCA with k-nearest neighbor (CCA-KNN) clustering, are reported. The best results are obtained using CCA-KNN which outperforms previous results on the Corel-5k and the ESP-Game datasets and achieves comparable results on the IAPRTC-12 dataset. In our experiments we evaluate CNN features in the existing models which bring out the advantages of it over dozens of handcrafted features. We also demonstrate that word embedding vectors perform better than binary vectors as a representation of the tags associated with an image. In addition we compare the CCA model to a simple CNN based linear regression model, which allows the CNN layers to be trained using back-propagation.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and indexing; I.2.10 [**Artificial Intelligence**]: Visual and Scene Understanding

Keywords

Image annotation, deep learning, word embeddings, CCA

1. INTRODUCTION

The metadata tags associated with images/videos are often used to search them. It is desirable to generate such tags automatically. Automatic image annotation is a labelling problem wherein the task is to predict multiple textual labels for an unseen image describing its contents or visual appearance [10, 2, 4, 13]. In these papers the goal is to predict a fixed number of tags for a given test image that accurately describe the visual content. Multiple features with the right type of model are shown to improve the annotation performance significantly in the current state of the art system [15, 4]. Yet, the dozens of handcrafted features serve as a bottleneck in designing scalable realtime systems. Hence, we use a feature (representing an image) from a Convolutional Neural Network (CNN) along with the word embedding vectors (representing the tags associated with it) in our proposed models. Here CNN features are extracted for images using a pre-trained VGG-16 [14] network, and the word embedding vector for a tag is extracted using a pre-trained skip-gram architecture (word2vec of [11]); both these networks are publicly available. This work, to the best of our knowledge, is the first attempt to use CNN features and word embedding vectors to solve the image annotation task and report results on all three standard datasets.

Our proposed models incorporate both CNN features and text features (word embedding vector), one set of models is based on the CCA and its variants and the other model is based on CNN regression. One variant, CCA-KNN significantly outperforms all the previously published results. We are able to achieve this without requiring any computationally expensive metric learning approaches as used by almost all successful models [13, 1, 12, 15]. Some papers using CCA for combing image and text have been previously proposed [3, 1], but the key differences are, we use CNN features as opposed to multiple handcrafted features (representing images) and we use word embedding vectors instead of binary vectors (representing tags). We show that the CNN feature outperforms multiple handcrafted features in [1].

In the CNN based model, the last layer of CaffeNet is replaced with a projection layer (to perform regression) and the resulting network is trained to map images to semantically meaningful word embedding vectors. This type of modeling has two advantages: firstly, it does not require dozens of handcrafted features; there is no need for metric learning or ways to combine those features efficiently. Secondly, the approach is substantially simpler to formulate than any other generative or discriminative models. In addition, we also provide the effectiveness of CNN features when used in other previously existing models.

2. PREVIOUS WORK

A large number of models have been proposed for automatic image annotation and retrieval. We discuss a few of the most successful ones. The models may be broadly

ICMR'15, June 23–26, 2015, Shanghai, China.

http://dx.doi.org/10.1145/2671188.2749391.

grouped into three groups - generative models, discriminative models and nearest neighbor based models. Examples of generative models are Cross Media Relevance Model (CMRM) [6], Continuous Relevance Model (CRM) and Multiple Bernoulli Relevance Model (MBRM) [2] . These may be viewed as mixture models or as nearest neighbor based approaches. They work by computing the joint probability of words and visual features. Given a test image, the model is used to compute conditional probability scores for words. Visual features are represented either by discretizing and clustering them or by using a kernel density estimate. The visual features are modeled using a multinomial distribution and the best results are obtained by modeling the words using a multiple Bernoulli distribution. Several successful nearest neighbor based models have been inspired by MBRM including Joint Equal Contribution (JEC) model [10], TagProp [4] and the 2PKNN model[15]. TagProp uses metric learning over a large number of features. The 2PKNN (two-pass KNN) technique finds semantic neighbors for each test image and the tags are predicted based on the weighted combination of distances. The optimal weights to combine base distances and features was determined via metric learning. Recently an attempt was made to improve the performance of CRM using Sparse Kernel Learning (SKL-CRM) [12] , in which they try to learn the optimal combination of kernels for the features.

3. FEATURE EXTRACTION

Here, we provide details about how the CNN features are extracted for images, followed by details about how to use word embedding vectors to represent the tags.

3.1 CNN features

Given an image, we extract a 4096-dimensional feature vector (X) using a pre-trained CNN on the ILSVRC-2012 dataset as described in Simonyan et al. [14]. We explored both VGG-16 and VGG-19 layered architecture features. Since both of them gave similar results, we used VGG-16. Features extracted from Caffe-Net provided by Caffe [7] (similar to AlexNet [9]) did not work as well as VGG-16, hence we used VGG-16 features for all our experiments. The features are computed by forward propagating a mean-subtracted 224x224 RGB image through eight convolutional layers and three fully connected layers. In our case, we resize all the images irrespective of their aspect ratio to 224x224 to make it compatible with the CNN.

3.2 Word embeddings

For each tag associated with an image, we represent the tag (word) by a 300 dimensional real valued feature vector using a Word2Vec tool and we call it as a word embedding vector ($E \in R^{l \times q}$), where l is the number of labels and $q = 300$ dimensions. These word vectors are obtained from a pre-trained skip-gram text modeling architecture introduced by Mikolov et al. [11]. It was shown that the model learns similar embedding vectors for semantically related words. Therefore, we use it to represent the annotations. We take the average of all the word embedding vectors (Y) associated with multiple tags representing an image. Formally, if there are k tags associated with an image I then $Y = \frac{1}{k}\sum_{i=1}^{k} E_i$ and their association is represented as $\{I, Y\}$. While reporting the result we refer to word embedding vectors as W2V.

3.3 Canonical Correlation Analysis (CCA)

Given a pair of views for an image – a visual feature (X, i.e., CNN feature) and the other the (Y, i.e., word embedding vector), CCA computes projections w_x and w_y for X and Y respectively to maximize their correlation. Concretely, for M samples, let $X \in R^{m \times p}$ and $Y \in R^{m \times q}$ be the two views of the data, then the projection vectors w_x *and* w_y are computed by maximizing the correlation coefficient ρ

$$\rho = \underset{w_x, w_y}{\arg\max} \frac{w_x^T X Y^T w_y}{\sqrt{(w_x^T X X^T w_x)(w_y^T Y Y^T w_y)}} \quad (1)$$

The dimensionality of these new basis vectors is less than or equal to the smallest dimensionality of the two variables. The canonical correlations are invariant to affine transformations of the variables. The solution is found by formulating it as a generalized eigenvalue problem [5]:

$$XY^T(YY^T)^{-1}YX^T w_x = \eta X X^T w_x, \quad (2)$$

where η is the eigenvalue corresponding to the eigenvector w_x. Multiple projectors can also be found which form a projection matrix $W_x \in R^{p \times l} \in$ and similarly $W_y \in R^{q \times l}$. In the case of regularized CCA, a term λI with $\lambda > 0$ is used to avoid overfitting.

3.4 Kernel CCA (KCCA)

Since CCA can only capture linear relationships, we propose to use a χ^2 kernel for exploiting the non linear relationships. The χ^2 kernel was found to be well suited in our experiments. The visual feature X is mapped to a high dimensional feature space $\mathcal{H}_x$ using a function ϕ_x. The ϕ_x mapping is achieved using a positive definite kernel function $K_x = \langle \phi_x, \phi_x \rangle \in R^{m \times m}$, where $\langle :, : \rangle$ is an inner product in $\mathcal{H}_x$. Similarly, the word embedding vector Y is mapped to $\mathcal{H}_y$ using the kernel function $K_y = \langle \phi_y, \phi_y \rangle \in R^{mxm}$. Kernel CCA finds the solution of w_x and w_y as a linear combination of the training data:

$w_x = \sum_{i=1}^{m} \alpha_i \phi_x(x_i)$ and $w_y = \sum_{i=1}^{m} \beta_i \phi_y(y_i)$. Since feature vector dimensions were high overfitting is an issue. To avoid this we used a regularized kernel CCA which finds $\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}$ by maximizing the following objective function:

$$\underset{\boldsymbol{\alpha}, \boldsymbol{\beta}}{\arg\max} \frac{\boldsymbol{\alpha}^T K_x K_y \boldsymbol{\beta}}{\sqrt{(\boldsymbol{\alpha}^T K_x^2 \boldsymbol{\alpha} + r_x \boldsymbol{\alpha}^T K_x \boldsymbol{\alpha})(\boldsymbol{\beta}^T K_y^2 \boldsymbol{\beta} + r_y \boldsymbol{\beta}^T K_y \boldsymbol{\beta})}} \quad (3)$$

The solution yields top l eigenvectors $\mathcal{W}_x = [\boldsymbol{\alpha}^1 \dots \boldsymbol{\alpha}^l]$ and $\mathcal{W}_y = [\boldsymbol{\beta}^1 \dots \boldsymbol{\beta}^l]$ which form the projection matrix.

3.4.1 Implementation details

CCA and KCCA with regularization was implemented as explained in [5]. Regularization was found to be important to avoid overfitting resulting in better performance. In the case of linear CCA, we project X onto W_x, project Y onto W_y and project E onto W_y:

$$U = (X - \mu_X)W_x \, , \, V = (Y - \mu_Y)W_y \, \text{and} \, Z = EW_y \quad (4)$$

Given a test image I_t, we extract deep learning visual features V_t and project it using w_x as $T = (V_t - \mu_X)W_x$ and compute the correlation distance to V. The corresponding tags associated with the closest matching V_i are assigned to the test image (tags are also ranked according to their frequency in the training dataset). If the tags are less than

the fixed annotation length then we pick the next closest match and transfer the tags, we repeat this until we obtain the required set of tags, in our case its five (to compare with previous work).

Similarly, in the case of KCCA, we kernelize X, Y and Z and later project onto $\mathcal{W}_x, \mathcal{W}_y, \mathcal{W}_x$ respectively. For a test image, we kernelize the visual features and follow the same procedure as above.

In CCA with KNN clustering (CCA-KNN) setup, after finding the correlation distance of T with V, we choose K semantic neighbor samples from each cluster (grouped according to its labels) for that particular test image and now their associated tags form a subset of tags Z_k (potential candidates for a test image). Later, we rank the words w for a test image I_t according to its probability score of:

$$P(I_t|w) = \sum_k \exp(-D(T, Z_k))1_k(w) \quad (5)$$

where, $D(T, Z_k)$ is the correlation distance between T and Z_k and $1_k(w)$ is an indicator function which takes a value 1 if the tag is present among neighbors and 0 otherwise.

3.5 CNN-based regression model

Inspired by the success of deep CNN architectures [9, 14]on the large scale image classification task, we use it to solve the task of automatic image annotation. To the best of our knowledge, this is the first attempt to formulate this problem based on a CNN The idea is to formulate the problem as a linear regression. We achieve this by replacing the last layer of Caffe-Net with a projection layer (fully connected layer) and we call it as a CNN regressor (CNN-R). CNN provides the mapping function which regresses the fixed size of the input image to a word embedding vector. For further architecture details, please refer to [9]. In this setup, we increased the learning rate for the newly introduced layer while reducing it for all the other previous layers and the reason being that we are trying to fine-tune the network previously trained on 1.2 million images. The input image size was fixed to be 227x227 and the final regressed output was a 300 dimensional vector. Since we have chosen to do linear regression, we use Euclidean loss (L2) instead of Softmax loss during the training phase. The prediction layer tries to predict the word embedding vector by minimizing the L2 loss depending on which, the model parameters are updated using Back propagation algorithm.

4. EXPERIMENTS

We evaluate on three standard publicly available image annotation datasets - Corel-5k [2], ESP-Game and IAPRC-12 [10]. These datasets contain a variety of images like natural scene, game, sketches, transportation vehicles, personal photos and so on, thus making it a challenging task.

Most papers use one type of metric, but there is a small number of papers which use other metrics (For instance, In [8] the precision and recall are computed **per word** but they are computed only for **non-zero recall** words and their average over all non-zero recall words are reported). We use the standard (most widely reported type of evaluation where the recall and precision are computed **per word** and their average over all the words are reported [13, 1, 12, 15, 10, 2]. We strictly adhere to computing the recall and precision **per word** (for all the words) and reporting their means over all the words, thus making it a fair comparison to the majority of the works in this area. Along with that, we also report **N+** denoting the number of words with non-zero recall value.

4.1 Experimental results and discussion

In order to have a fair comparison with the previously reported results, we follow the same train and test split as reported in [4] and also fix the length of the annotations (five tags) for a test image. In the first subsection, we evaluate the CNN features with our proposed model and compare its performance with all other previous work. We also provide the results of using some existing models, 2PKNN, JEC and SVM-DMBRM, with the new set of CNN features. This helps in understanding how well the CNN features perform against 15 handcrafted features (local + global).

In the second subsection, we evaluate the importance of the word embedding vectors. We do this by picking the best performing model in Table 1 and use word embedding vectors instead of frequency counts to represent the tags. Since each image is annotated with a unique set of tags in all three datasets, the frequency vector representing the tags of an image just turns out be a simple binary vector (BV).

4.1.1 Evaluation of CNN features

Results are presented in the Table 1. We see that our proposed CCA-KNN[1] model outperforms all other previous work on Corel-5k and ESP-Game datasets. More precisely, our method provides 2.6% and 13.5% increase in F1 measure on Corel-5k and ESP-Game datasets respectively over the next best method. Also we are better in terms of N+ (# non-zero recall) measure, which is a clear indication that our method generalizes well to unseen test images. In the case of IAPRTC-12 dataset, our model yields comparable results to the state-of-the-art.

We ran CNN features with some of the other models in the literature (JEC, 2PKNN and SVM-DBRM). In the case of 2PKNN the results were worse than using all 15 feature vectors while for the other two models the results were better or comparable to using their original set of features. This indicates that CNN features are powerful and are a good candidate for replacing a large set of features.

4.1.2 Importance of word embeddings

Table 1 also shows that word embedding vectors (W2V) work better than binary vectors (BV) with our best performing model CCA-KNN. This suggests that word embedding vectors provide a better representation for words than their binary form – presumably because semantically related words tend to have similar word embedding vectors.

4.1.3 Evaluation of CNN-R model

Experimental evaluation results of our CNN-R models on all three datasets are provided in Table 1 with comparison to some of the best performing models in the literature. From Table 1, we see that CNN-R outperforms many previous models but not the CCA models with CNN features and is competitive with the the 2PKNN model and Tagprop. CNN-R has a clear advantage over all the existing methods for the following reasons: no need to extract multiple low level features and to incorporate high level semantics; no metric learning; we can fine-tune the deep architecture even

[1]Qualitative result analysis at http://goo.gl/wS4jSl

Method	Feature		Corel-5K				ESP Game				IAPRTC-12			
	Visual	text	P	R	F	N+	P	R	F	N+	P	R	F	N+
JEC [10]	HC	-	27	32	29	139	22	25	23	224	28	29	29	250
MBRM [2]	HC	-	24	25	25	122	18	19	19	209	24	23	24	223
TagProp(σML) [4]	HC	-	33	42	37	160	39	27	32	239	46	35	40	266
2PKNN [15]	HC	-	39	40	39.5	177	51	23	31.7	245	49	32	38.7	274
2PKNN+ML [15]	HC	-	44	46	45	191	53	27	36	252	54	37	**44**	278
SVM-DMBRM [13]	HC	-	36	48	41	197	**55**	25	34	259	56	29	38	**283**
KCCA-2PKNN [1]	HC	-	42	46	44	179	-	-	-	-	**59**	30	40	259
SKL-CRM [12]	HC	-	39	46	42	184	41	26	32	248	47	32	38	274
JEC	VGG-16	-	31	32	32	141	26	22	24	234	28	21	24	237
2PKNN	VGG-16	-	33	30	32	160	40	23	29	250	38	23	29	261
SVM-DMBRM	VGG-16	-	42	45	43	186	51	26	35	251	58	27	37	268
CCA	VGG-16	W2V	35	46	40	172	29	32	30	250	33	32	33	268
KCCA	VGG-16	W2V	39	**53**	45	184	30	36	33	252	38	**39**	38	273
CCA-KNN	VGG-16	BV	39	51	44	192	44	32	37	254	41	34	37	273
CCA-KNN	VGG-16	W2V	**42**	52	**46**	**201**	46	**36**	**41**	**260**	45	38	41	278
CNN-R	Caffe-Net	W2V	32	41.3	37.2	166	44.5	28.5	34.7	248	49	31	37.9	272

Table 1: Experimental results of our proposed models with previously reported best scores on all three datasets. P: Average Precision, R: Average Recall, N+: Number of distinct words that are correctly assigned to at least one test image.

for a small dataset and this type of model is also capable of predicting new set of previously unseen classes with the help of word embeddings vectors. We believe that the performance can be improved further with some regularization.

5. CONCLUSION

We have explored CNN features and word embedding vectors for image annotation task and have introduced some new models to take advantage of both these features. Empirically, one of our proposed model CCA-KNN yields better results when compared to previous work. We also validated the advantage of using CNN features over 15 handcrafted features for some existing models and showed that performance is often comparable to their previously reported results. CNN features avoids the use of computing multiple engineered features and also the computationally expensive process of metric learning (a trend in most recent papers). We also introduce a simple and efficient way of formulating the image annotation problem as a CNN based regressor which may be very useful in the real world applications.

6. ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant #IIS-0910884. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

7. REFERENCES

[1] L. Ballan, T. Uricchio, L. Seidenari, and A. Del Bimbo. A cross-media model for automatic image annotation. In *ICMR*, page 73, 2014.

[2] S. L. Feng, R. Manmatha, and V. Lavrenko. Multiple bernoulli relevance models for image and video annotation. In *CVPR'04*, pages 1002–1009, 2004.

[3] Y. Gong, Q. Ke, M. Isard, and S. Lazebnik. A multi-view embedding space for modeling internet images, tags, and their semantics. *IJCV*, 106(2):210–233, 2014.

[4] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *In ICCV*, 2009.

[5] D. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664, 2004.

[6] J. Jeon, V. Lavrenko, and R. Manmatha. Automatic image annotation and retrieval using cross-media relevance models. In *SIGIR '03*, pages 119–126, 2003.

[7] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, pages 675–678, 2014.

[8] M. M. Kalayeh, H. Idrees, and M. Shah. Nmf-knn: Image annotation using weighted multi-view non-negative matrix factorization. In *CVPR'14*, 2014.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[10] A. Makadia, V. Pavlovic, and S. Kumar. A new baseline for image annotation. In *ECCV '08*, pages 316–329, 2008.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[12] S. Moran and V. Lavrenko. A sparse kernel relevance model for automatic image annotation. *ntl. Journal of Multimedia Information Retrieval*, 3(4):209–229, 2014.

[13] V. N. Murthy, E. F. Can, and R. Manmatha. A hybrid model for automatic image annotation. In *ICMR'14*, pages 369:369–369:376, 2014.

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[15] Y. Verma and C. V. Jawahar. Image annotation using metric learning in semantic neighborhoods. In *ECCV'12*, pages 836–849, 2012.