INNOVATION FOR DESCRIPTION MAPPING E- MEDIA DATABASES MATCHING NAME DOMAIN IN SOCIAL NETWORK CONTEXT USING SOFTWARE PROCESS MODEL FOR PHOTO AUTO TAGGING

Mr.Pankaj Agarkar,
Computer Engg. PhD Research Scholar
JJTUniversity , Jhunjhunu , Rajastan , India

Dr.S.D.Joshi
Computer Engg. PhD Research Guide
JJTUniversity , Jhunjhunu, Rajastan India ,

ABSTRACT
Most personal photos that are shared online are embedded in some form of social network, and these social networks are a potent source of contextual information that can be leveraged for automatic image understanding. In this paper, we investigate the utility of social network context for the task of automatic face recognition in personal photographs. We combine face recognition scores with social context in a conditional random field (CRF) model and apply this model to label faces in photos from the popular online social network Face book, which is now the top photo-sharing site on the Web with billions of photos in total. We demonstrate that our simple method of enhancing face recognition with social network context substantially increases recognition performance beyond that of a baseline face recognition system.
Software systems come and go through a series of passages that account for their inception, initial development, productive operation, upkeep, and retirement from one generation to another. This article categorizes and examines a number of methods for describing or modeling how software systems are developed. It begins with background and definitions of traditional software life cycle models that dominate most textbook discussions and current software development practices. This is followed by a more comprehensive review of the alternative models of software evolution that are of current use as the basis for organizing software engineering projects and technologies. Software design is the process of implementing software solutions to one or more set of problems. One of the important parts of software design is the software requirements analysis (SRA). It is a part of the software development process that lists specifications used in software engineering. If the software is "semi-automated" or user centered, software design may involve user experience design yielding a storyboard to help determine those specifications. If the software is completely automated (meaning no user or user interface), a software design may be as simple as a flow chart or text describing a planned sequence of events. There are also semi-standard methods like Unified Modeling Language and Fundamental modeling concepts. In either case, some documentation of the plan is usually the product of the design. Furthermore, a software design may be platform-independent or platform-specific, depending upon the availability of the technology used for the design.

KEYWORDS: face naming, social network, unconstrained web videos mining, unsupervised, software life cycle concept.

INTRODUCTION

A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. A descriptive model describes the history of how a particular software system was developed. Descriptive models may be used as the basis for understanding and improving software development processes or for building empirically grounded prescriptive models (Curtis, Krasner, Iscoe, 1988). A prescriptive model prescribes how a new software system should be developed. Prescriptive models are used as guidelines or frameworks to organize and structure how software development activities should be performed, and in what order.
Typically, it is easier and more common to articulate a prescriptive life cycle model for how software systems should be developed. This is possible since most such models are intuitive or well reasoned. This means that many idiosyncratic details that describe how a software system is built in practice can be ignored, generalized, or deferred for later consideration. This, of course, should raise concern for the relative validity and robustness of such life cycle models when developing different kinds of application systems, in different kinds of development settings, using different programming languages, with differentially skilled staff, etc. However, prescriptive models are also used to package the development tasks and techniques for using a given set of software engineering tools or environment during a development project.

Descriptive life cycle models, on the other hand, characterize how particular software systems are actually developed in specific settings. As such, they are less common and more difficult to articulate for an obvious reason: one must observe or collect data throughout the life cycle of a software system, a period of elapsed time often measured in years. Also, descriptive models are specific to the systems observed and only generalizable through systematic comparative analysis. Therefore, this suggests the prescriptive software life cycle models will dominate attention until a sufficient base of observational data is available to articulate empirically grounded descriptive life cycle models.

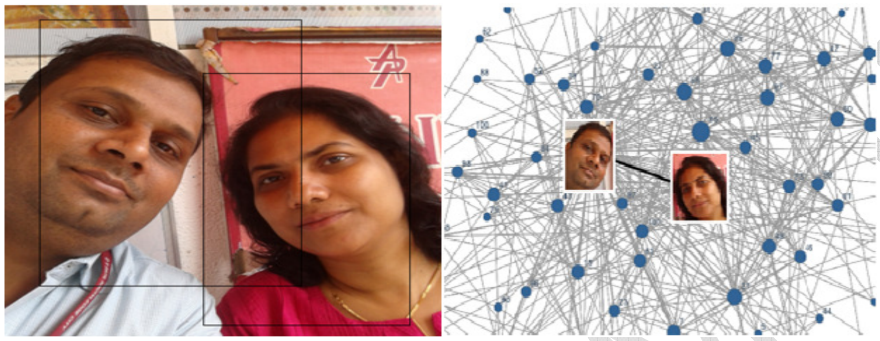

Figure 1: Shows social network context

An increasing number of personal photographs are uploaded to online social networks, and these photos do not exist in isolation. Each shared image likely arrives in a batch of related photos from a trip or event; these are then associated with their photographer and broadcast out to that photographer's hundreds of online friends, and they join a collection of billions of other photographs, some of which have been manually labeled with the people they contain and other information. Social networks are an important source of image annotations, and they also provide contextual information about the social interactions among individuals that can facilitate automatic image understanding. Despite the large and growing number of images that are embedded in online social networks, surprisingly little is known about the utility of social context for automatically parsing images. In this paper, we take a first step in this direction; we focus on the specific problem of automatic face recognition in personal photographs. We show that social network context can help tremendously even when it is Figure 1. A typical hand-tagged photo from the Face book social network with all tagged regions superimposed on the image. The visualization below the photo illustrates the social connections among these two individuals and their friends; for clarity, only a sample of the full graph is shown. These friendship links and other social network context can boost the accuracy of automatic face recognition in new photographs.

LITERATURE SURVEY

We investigate the problem of face identification in broadcast programs where people names are obtained from text overlays automatically processed with Optical Character Recognition (OCR) and further linked to the faces throughout the video. To solve the face-name association and propagation, we propose a novel approach that combines the positive effects of two Conditional Random Field (CRF) models: a CRF for person idolization (joint temporal segmentation and association of voices and faces) that benefit from the combination of multiple cues including as main contributions the use of identification sources (OCR appearances) and recurrent local face visual identification of the person clusters that improves identification performance thanks to the background (LFB) playing the role of a named ness feature a second CRF for the joint is of further idolization statistics.[1].
We describe a probabilistic method for identifying characters in TV series or movies. We aim at labeling every character appearance and not only those where a face can be detected. Consequently, our basic unit of appearance is a person track (as opposed to a face track). We evaluate our approach on the first 6 episodes of The Big Bang Theory and achieve an absolute improvement of 20% for person identification and 12% for face recognition.[2]
In modern face recognition, the conventional pipeline consists of four stages: detect⇒align⇒represent⇒classify. We revisit both the alignment step and the representation step by employing explicit 3D face modeling in order to apply a piecewise affine transformation, and derive a face representation from a nine-layer deep neural network. This deep network involves more than 120 million parameters using several locally connected layers without weight sharing, rather than the standard convolutional layers. Our method reaches an accuracy of 97.35% on the Labeled Faces in the Wild (LFW) dataset, reducing the error of the current state of the art by more than 27%, closely approaching human-level performance.[3]

CLASSIC SOFTWARE LIFE CYCLE MODELS

These classic software life cycle models usually include some version or subset of the following activities:[8]

A. **SYSTEM INITIATION/PLANNING:** where do systems come from? In most situations, new feasible systems replace or supplement existing information processing mechanisms whether they were previously automated, manual, or informal.

B. **REQUIREMENT ANALYSIS AND SPECIFICATION:** identifies the problems a new software system is suppose to solve, its operational capabilities, its desired performance characteristics, and the resource infrastructure needed to support system operation and maintenance.

C. **FUNCTIONAL SPECIFICATION OR PROTOTYPING:** identifies and potentially formalizes the objects of computation, their attributes and relationships, the operations that transform these objects, the constraints that restrict system behavior, and so forth.

D. **PARTITION AND SELECTION** (Build vs. Buy vs. Reuse): given requirements and functional specifications, divide the system into manageable pieces that denote logical subsystems, then determine whether new, existing, or reusable software systems correspond to the needed pieces.

E. **ARCHITECTURAL DESIGN AND CONFIGURATION SPECIFICATION:** defines the interconnection and resource interfaces between system subsystems, components, and modules in ways suitable for their detailed design and overall configuration management.

F. **DETAILED COMPONENT DESIGN SPECIFICATION:** defines the procedural methods through which the data resources within the modules of a component are transformed from required inputs into provided outputs.

G. **COMPONENT IMPLEMENTATION AND DEBUGGING:** codifies the preceding specifications into operational source code implementations and validates their basic operation.

H. **SOFTWARE INTEGRATION AND TESTING:** affirms and sustains the overall integrity of the software system architectural configuration through verifying the consistency and completeness of implemented modules, verifying the resource interfaces and interconnections against their specifications, and validating the performance of the system and subsystems against their requirements.

I. **DOCUMENTATION REVISION AND SYSTEM DELIVERY:** packaging and rationalizing recorded system development descriptions into systematic documents and user guides, all in a form suitable for dissemination and system support.

J. **DEPLOYMENT AND INSTALLATION:** providing directions for installing the delivered software into the local computing environment, configuring operating systems parameters and user access privileges, and running diagnostic test cases to assure the viability of basic system operation.

K. **TRAINING AND USE:** providing system users with instructional aids and guidance for understanding the system's capabilities and limits in order to effectively use the system.

L. **SOFTWARE MAINTENANCE:** sustaining the useful operation of a system in its host/target environment by providing requested functional enhancements, repairs, performance

DESIGN CONSIDERATIONS

There are many aspects to consider in the design of a piece of software. The importance of each should reflect the goals the software is trying to achieve. Some of these aspects are:[7].

A. **COMPATIBILITY** - The software is able to operate with other products that are designed for interoperability with another product. For example, a piece of software may be backward-compatible with an older version of itself.

B. **EXTENSIBILITY** - New capabilities can be added to the software without major changes to the underlying architecture.

C. **FAULT-TOLERANCE** - The software is resistant to and able to recover from component failure.

D. **MAINTAINABILITY** - A measure of how easily bug fixes or functional modifications can be accomplished. High maintainability can be the product of modularity and extensibility.

E. **MODULARITY** - the resulting software comprises well defined, independent components which lead to better maintainability. The components could be then implemented and tested in isolation before being integrated to form a desired software system. This allows division of work in a software development project.

F. **RELIABILITY** - The software is able to perform a required function under stated conditions for a specified period of time.

G. **REUSABILITY** - parts or all of the software can be used in other projects with no, or only slight, modification.

H. **ROBUSTNESS** - The software is able to operate under stress or tolerate unpredictable or invalid input. For example, it can be designed with resilience to low memory conditions.

I. **SECURITY** - The software is able to withstand hostile acts and influences.

J. **USABILITY** - The software user interface must be usable for its target user/audience. Default values for the parameters must be chosen so that they are a good choice for the majority of the users.

K. **PERFORMANCE** - The software performs its tasks within a user-acceptable time. The software does not consume too much memory.

L. **PORTABILITY** - The usability of the same software in different environments.

M. **SCALABILITY** - The software adapts well to increasing data or number of users.

MODULE DESCRIPTION

A. **FACE SEQUENCE EXTRACTION:** This modules consists of 2 parts

B. **FRAMES EXTRACTION:** The frames are extracted from video.

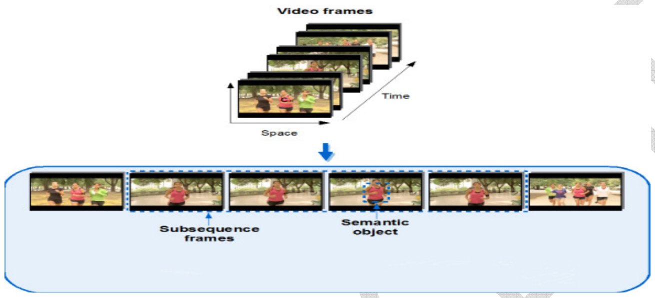

Fig.2 frames Extraction

a) FACE SEQUENCE ASSOCIATION: Frames that contains images are considered. From that frames faces are extracted.The recognition of human faces is not so much about face recognition at all it is much more about face detection! It has been proven that the first step in automatic facial recognition – the accurate detection of human faces in arbitrary scenes, is the most important process involved.

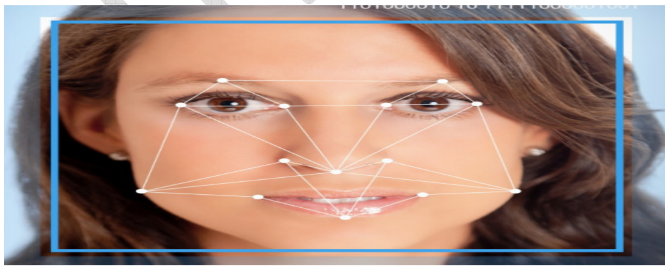

Fig.2 Face sequence association

C. **2. VIDEO CAPTION RECOGNIZING:** video name is also considered as a part of face identification. Although it does not play a integral part but yet it can help in sometimes.

D. **3. FACE NAME ASSOCIATION:** This part is the database part in the project. Here all the faces and its corresponding geometry features are stored.

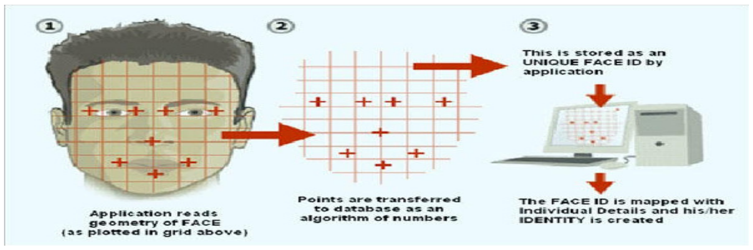
1
2
3
This is stored as an UNIQUE FACE ID by application
Application reads geometry of FACE (as plotted in grid above)
Points are transferred to database as an algorithm of numbers
The FACE ID is mapped with individual Details and his/her IDENTITY is created

Fig.3 Face name association

While undergoing the face- name association, following factors are considered

a) **LIGHTING COMPENSATION:** This will adjust contrast of image.

b) **EIGEN FACES:** Face Images are projected into a feature space ("Face Space") that best encodes the variation among known face images. The face space is defined by "Eigenfaces" which are the eigenvectors of the set of faces.

FACE TO NAME RETRIEVAL

A. **SKIN COLOR EXTRACTION:** After getting frames skin-tone color is extracted from the input image as the most important information of human face. Further coarse detection, edge extraction and blurring are executed for skin-tone flag map. This emphasizes the edges of skin-tone for face candidate detection.

B. **FACE JUDGMENT:** After lines-of-face detection, there may be some remaining noises because the lines-of-face template can only detect skin-tone contour.

C. **TEMPLATE MATCHING:** The matched template will be used compares with face name association.And the corresponding name will be considered.

CONCLUSION AND FUTURE WORK

We have presented the modeling of multiple relationshipsusing CRF for celebrity naming in the Web video domain. In view of the incomplete and noisy metadata, Conditional Random Field (CRF) softly encodes these relationships while allowing null assignments by considering the uncertainty in labelling. Experimental results basically show that these nice properties lead to performance superiority over several existing approaches. The consideration of between-video relationships also results in further performance boost, mostly attributed to the capability of rectifying the errors due to missing names and persons.

The price of improvement, nevertheless, also comes along with increase in processing time and the number of false positives. Fortunately, the proposals of leveraging social relation and joint labeling by sequential video processing still make CRF scalable in terms of speed and memory efficiency. While the overall performance of the proposed approach is encouraging, the effectiveness is still limited by facial feature similarity, which is used in the unary energy term and pairwise visual relationship. It can be further enhanced by considering different parameters like image background and other parameters for providing better description.

REFERENCES

[1] S. Satoh, Y. Nakamura, and T. Kanade, "Name-It: Naming and detecting faces in news videos," IEEE Multimedia, vol. 6, no. 1, pp. 22–35, Jan.–Mar. 1999.
[2] Y. F. Zhang, C. S. Xu, H. Q. Lu, and Y. M. Huang, "Character identification in feature-length films using global face-name matching," IEEE Trans. Multimedia, vol. 11, no. 7, pp. 1276–1288, Nov. 2009.
[3] Z. Stone, T. Zickler, and T. Darrell, "Toward large-scale face recognition using social network context," Proc. IEEE, vol. 98, no. 8, pp.1408–1415, Aug. 2010.
[4] J. Choi, W. De Neve, K. N. Plataniotis, and Y. M. Ro, "Collaborativeface recognition for improved face annotation in personal photo collections shared on online social networks," IEEE Trans. Multimedia,vol. 13, no. 1, pp. 14–28, Feb. 2011.
[5] G. Paul, K. Elie, M. Sylvain, O. Marc, and D. Paul, "A conditional random field approach for face identification in broadcast news using overlaid text," in Proc. IEEE Int. Conf. Image Process., Oct. 2014, pp. 318–322.
[6] J. S. Yedidia, W. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," IEEE Trans. Inf. Theory, vol. 51, no. 7, pp. 2282–2312, Jul. 2005.
[7] Suryanarayana, Girish (November 2014). Refactoring for Software Design Smells. Morgan Kaufmann. p. 258. ISBN 978-0128013977. Retrieved 31 January 2015.
[8] Ambriola, V., R. Conradi and A. Fuggetta, Assessing process-centered software engineering environments, ACM Trans. Softw. Eng. Methodol. 6, 3, 283-328,1997.