Leveraging High-level and Low-level Features for Multimedia Event Detection

Lu Jiang, Alexander G. Hauptmann, Guang Xiang
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
{lujiang,alex,guangx}@cs.cmu.edu

ABSTRACT

This paper addresses the challenge of Multimedia Event Detection by proposing a novel method for high-level and low-level features fusion based on collective classification. Generally, the method consists of three steps: training a classifier from low-level features; encoding high-level features into graphs; and diffusing the scores on the established graph to obtain the final prediction. The final prediction is derived from multiple graphs each of which corresponds to a high-level feature. The paper investigates two graph construction methods using logarithmic and exponential loss functions, respectively and two collective classification algorithms, i.e. Gibbs sampling and Markov random walk. The theoretical analysis demonstrates that the proposed method converges and is computationally scalable and the empirical analysis on TRECVID 2011 Multimedia Event Detection dataset validates its outstanding performance compared to state-of-the-art methods, with an added benefit of interpretability.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms, Experimentation, Performance

Keywords

Collective classification, feature fusion, multi-modal integration

1. INTRODUCTION

In the past decade, the Internet has witnessed an explosion of multimedia content. Comparable to the days in the late 1990s, when people usually got lost in the rising sea of web pages, now they are overwhelmed by the vast amounts of multimedia content on the web. To advance the development of new technologies for content understanding, the NIST TRECVID effort initiated a new task called Multimedia Event Detection(MED) in 2010. The task is more challenging than the previous ones such as object recognition and action detection, since an event is often more difficult to be characterized and thus to be detected. An event is "an activity-centered happening that involves people engaged in process-driven actions with other people and/or objects at a specific place and time". For example, the event "attempting a board trick" includes video clips such as skating, skiing, surfing and even finger skating. Because of the inherent difficulties of this task, all research tries to exploit multiple types of features to achieve better performance. Generally these feature can be divided into two categories, namely high-level and low-level features. Low-level features such as SIFT and STIP[1], capture the local appearance and texture statistics of objects in the video represented by a collection of interest points. On the other hand, high-level features are represented by a real number estimating the probability of observing a concept in the video. Detection based on high-level features is more consistent with human's understanding and reasoning about the task, where an event is characterized by the presence/absence of certain concepts rather than interest points. E.g. a video is probably about "birthday party" if visual concepts such as "birthday cake", "faces" and the audio concept "cheering" are present. Therefore, high-level features or attributes are favored in the event and scene detection in many studies[2, 3].

The process of combining multiple modalities is called fusion, which can be achieved either by combing the feature vectors into a single long vector based on which a model is trained, namely early fusion[4], or by learning different models for different modalities and aggregating the outputs by different models, usually called late fusion[5]. Many studies [3, 6, 7] have shown that fusing features often leads to better performance as features are usually complementary. Current fusion approaches offer a simple yet effective way for combination of multiple modalities. However, the major problem, especially for the early fusion, is the lack of interpretability of the reason for the score change during the fusion. Therefore it is difficult for a human to understand and trust the fusion result. To tackle this problem, in this paper, we propose a method called Feature Fusion by Collective Classification(FFCC). Our work is motivated by the successful work in classification on social networks [8] and collaboration networks[9]. The intuition behind the method is straightforward and can be summarized into the following process: first we train a classifier we call a local classifier with low-level features to capture the general idea of the events in the video

MM 12, October 29–November 2, 2012, Nara, Japan.

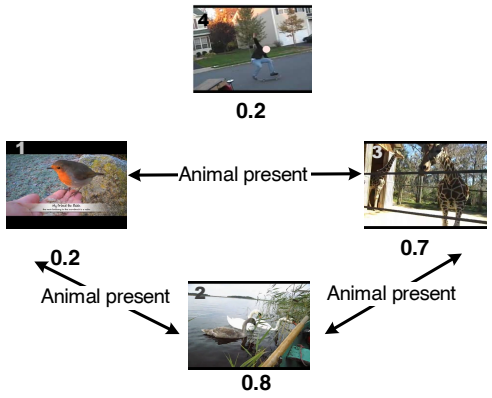


Figure 1: An illustrative graph on high-level feature "animal". The video ID is presented at the left-top of each video and the number below indicates the score generated by the local classifier.

clips; then we construct a set of graphs, one graph for each high-level feature, in which two video clips are linked if the high-level feature concept is either present or absent in both clips, see Fig. 1. Finally, the event scores obtained from the local classifier are diffused through the graphs to obtain the final prediction. The probability diffusion is achieved with a collective classification algorithm [10], such that the final score of all video clips are inferred simultaneously, by updating a video's score according to the scores of its neighbors. Unlike other problems on social network, which only involve a single graph, in this paper, we consider a video clip connected through multiple graphs, each of which corresponds to a high-level concept. The reason for adopting the multiple graphs representation is its interpretability. Leveraging the graph structure, FFCC can correct the false positive connected with true negatives, or false negative connected with true positives. For example, suppose we are detecting the event "feeding an animal" using the graph in Fig. 1, the score of video 1 will be boosted during collective classification because it receives support from scores of video clip 2 and 3. The score of video 4, on the other hand, stays unchanged as it is isolated from the concept graphs of the other videos.

Compared to early and late fusion methods, FFCC has the following three benefits: first, it is a probabilistic approach thus has a solid theoretical background; secondly, the fusion result is interpretable. Since each edge in the graphs carries semantic meaning, the score changes during the collective classification can be understood and validated or verified through inspection. In addition, FFCC represents a good means to incorporate prior knowledge into the graph, which is especially useful for ad-hoc event detection where the training samples are extremely scarce.

In summary, the contributions of this paper are twofold:

- We introduce a novel feature fusion method based on collective classification on multiple graphs. The empirical analysis demonstrates that the proposed method improves not only the performance but also the interpretability over the state-of-the-art fusion methods.
- We propose to use Markov Random Walk as an alternative collective classification algorithm and prove its convergence in our framework.

The rest of this paper is organized as follows: Section 2 gives a brief introduction of related work. Section 3 presents FFCC in details. The experimental results are discussed in Section 4 and conclusions are presented in the final section.

2. RELATED WORK

The idea of feature fusion has been broadly applied in various tasks in multimedia retrieval and classification. Snoek et al. propose the concept of early and late fusion and empirically evaluate their performance on the TRECVID video retrieval benchmark [3]. A successful application of early fusion is the spatial pyramid matching[11, 12], in which each image is divided into tiles and features extracted from different tiles are concatenated to form the final feature vector. By fusing the features from different tiles, it encodes the spatial information about the image. In terms of the late fusion, Kludas et al. claims that maximal performance will be obtained while fusing independent modalities[6]. However, independent modalities are rare in practice and thus many researchers concentrate on studying the statistical dependency between modalities. Wu et al. propose a method based on Principle Component Analysis (PCA) and Independent Component Analysis (ICA) to exploit the dependency between different modalities[13]. However, since the method is based on PCA, it assumes 1) that features with large variances are important and 2) that the principal components are orthogonal. As these assumption is too restrictive to hold in many data sets, Rasiwasia el al.[14] propose to eliminate the dependency by Canonical Correlation Analysis (CCA) i.e. projecting the original feature spaces into a so-called semantic space that maximizes the correlation between different modalities. As CCA only requires a linear relationship between the variable in different modalities, the assumptions of Wu's method are relaxed. Compared to Wu's idea, our paper can be regarded as an attempt to study the non-linear correlation in modality fusion.

Collective classification have been drawing increasing attention in the machine learning community because of its outstanding classification performance on network data such as the Web[15], social networks [8] and collaboration networks[10, 9]. Its distinguishing property, compared with supervised and unsupervised learning, is that the independent and identical distribution assumption can be ignored. The most dominant collective classification methods are Loopy Belief Propagation and Gibbs Sampling. In this paper, we investigate Markov random walk as an alternative collective classification algorithm.

3. MODALITIES FUSION

In this section, we present the FFCC algorithm. Specifically, Section 3.1 introduces some intuitions and the high-level algorithm. The two key issues of graph construction and collective classification are discussed in Section 3.2 and Section 3.3. Finally, the computational complexity is analyzed in the last section.

3.1 Overall Algorithm

First of all, we formulate our intuition as a binary classification problem. Suppose we are given a set of training samples consists of $X_{tr} = \{x_i\}_{i=1}^N$, $Z_{tr} = \{z_i\}_{i=1}^N$ ($x_i \in \mathbb{R}^{M_L}$, $z_i \in \mathbb{R}^{M_H}$) and their labels $Y_{tr} = \{y_i\}_{i=1}^N$ ($y_i \in \mathcal{Y} = \{-1, +1\}$) where x_i,z_i represents the low-level and high-level feature

vector for the i^{th} sample in the training set, respectively. Similarly, test samples for high-level and low-level features are represented by $X_{ts} = \{x_i\}_{i=1}$ and $Z_{ts} = \{z_i\}_{i=1}$. Our goal is to learn a final hypothesis h_f for all test samples. For the convenience, we will use the subscript to index sample and superscript to index the feature dimension. E.g. x_i^j denotes the value of j^{th} dimension of the i^{th} sample.

We define a graph $G = (V, E)$, where the vertex set encompasses all samples in both training and test set, accordingly $V = V_{tr} \cup V_{ts}$. Each edge e_{ij}, which connects vertex v_i and v_j, is associated with a weight w_{ij} measuring the intensity of the connection. In this paper we only consider undirected graphs i.e. for each edge $w_{ij} = w_{ji}$.

A local hypothesis h_l denotes the real valuated function learned from low-level features which estimates the posterior probability of the label conditioned on low-level features. We call it local because it inherits the i.i.d. assumption that the data is independent and identically distributed. Most classification algorithms that follow this assumption, e.g. logistic regression and SVM, can be plugged in to learn a local hypothesis. However, for a non-probabilistic model such as SVM, a calibration may be useful to convert the scores into a posterior probability. In contrast to the local hypothesis, a relational hypothesis h_r is a posterior probability learned by collective classification algorithms, in which i.i.d. is no longer expected to hold and the probability of a sample's label is affected by those of its neighbors in graph G.

Algorithm 1: Overview of the Feature Fusion by Collective Classification

1 train a local hypothesis h_l using low-level training samples (X_{tr}, Y_{tr});
2 **foreach** $\mathbf{z}^i$ **do**
  construct a graph G_i;
  train a relational hypothesis h_{r_i};
end
3 return the optimal final hypothesis found in the training set;

Given the above notations, the proposed method can be summarized in the following three steps listed in Algorithm 1. First the algorithm obtains a local hypothesis h_l from a local classifier; then it constructs a graph for each high-level feature $\mathbf{z}^i$, on which a relational hypothesis h_{r_i} is trained using a collective classification algorithm; finally, it searches the best hypothesis over the training set and returns it as the final hypothesis h_f. For instance, we may train a SVM classifier on low-level features in Step 1; then in Step 2 a set of graphs are constructed by high-level features and the SVM prediction scores are updated by the collective classification algorithm on the established graphs. Finally, the optimal scores are selected from the updated ones as the final hypothesis.

If we assume the graphs are mutually independent, the final hypothesis can be calculated from (see Appendix for the derivation):

$$\log h_f = \sum_{z^i \in S} \log h_{r_i} - (|S| - 1) \log h_l \tag{1}$$

where S is a subset of high-level features. If the graphs are dependent, the average of the relational hypothesis can be applied to estimate the final hypothesis. Sometimes when the number of training samples is insufficient, we may need to smooth the final hypothesis with the local hypothesis:

$$\tilde{h}_f = \frac{1-\alpha}{h_l^{|S|-1}} \prod_{z^i \in S} h_{r_i} + \alpha h_l \tag{2}$$

The first part of the right hand side can be regard as the weighted average of all relational hypothesis. Eq. 2 models the final hypothesis h_f as a linear interpolation of the relational hypothesis with the local hypothesis with the coefficient α. Smoothing is often applied in the case where training samples are insufficient to obtain an accurate hypothesis. Notice that in both Eq. 1 and Eq. 2 only a subset of features that are relevant to the label contribute to the final hypothesis. Searching the global optimal features subset is an NP-hard problem, consequently in Step 3 we apply a forward wrapper with hill-climbing search to get the subset of features resulting in a locally optimal final hypothesis. In the following two subsections, we will elaborate the non-trivial methods in Step 2 by answering two questions: how to construct a graph from training data and how to train a relational hypothesis?

3.2 Graph Construction

The goal of graph construction is to construct "homophilous" graphs using high-level features, in which the samples with same labels tends to be linked with higher weights than those with different labels. In this paper, we concentrate on graph construction using individual high-level feature, i.e. the connection between two samples solely depends on a single high-level feature, say the k^{th} feature $\mathbf{z}^k$. Two video clips are connected if in which the corresponding high-level concept is present (or similarly defined as absent) in both clips. For each vertex $v_i, v_j \in \mathbf{V}$ the edge weight is calculated from the following equation.

$$w_{ij} = \begin{cases} |z_i^k - z_j^k| & (z_i^k - \delta)(z_j^k - \delta) > 0 \\ \text{disconnected} & \text{otherwise} \end{cases} \tag{3}$$

where δ is a threshold determined by the training set (Z_{tr}, Y_{tr}). A plausible way of learning a reasonable δ is through mutual information:

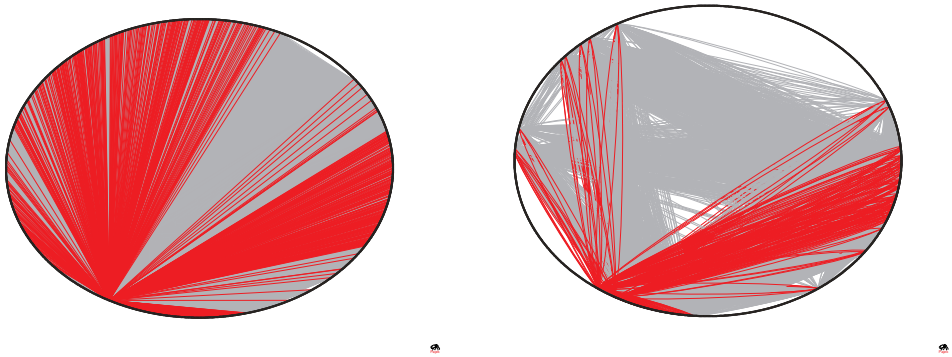
$$\delta = \arg\max_{\delta} H(Y_{tr}) - H(Y_{tr}|Z_{tr}^k; \delta) \tag{4}$$

where $H(Y_{tr})$ and $H(Y_{tr}|Z_{tr}^k; \delta)$ are entropy and conditional entropy of the label. The optimal δ is obtained when the correlation between the feature and the label is maximized. In other words, the amount of information on label is maximized using the high-level feature with the optimal δ.

Mutual information offers an intuitive yet aggressive way for the graph construction. We call it aggressive because it preferentially exploits more potential edges between vertices. In contrast, we may conceive a cautious way which selects the parameter δ that minimizes the following loss function:

$$\delta = \arg\min_{\delta} \sum_{z_i \in \mathbf{Z_{tr}}} e^{\epsilon(z_i^k; \delta)} - \sum_{i=1}^{N} I((z_i^k - \delta) y_i > 0) \tag{5}$$

where $\epsilon(z_i^k; \delta) = \sum_{i=1}^{N} I((z_i^k - \delta) y_i < 0)$ and $I(\cdot)$ is the indicator function equaling 1 when $(z_i^k - \delta) y_i < 0$, 0 otherwise. In the above loss function, $\epsilon(z_i^k; \delta)$ counts the number of incorrectly connected vertices in the graph and penalizes them by the exponential loss. The loss attains minima if



(a) log-loss (b) exp-loss

Figure 2: The graph generated by log and exp loss function on feature "walk/running" in Event "Parkour"

the high-level feature perfectly separates the training vertices into two components with all positive samples residing in one set and all negative samples in the other, otherwise the loss increases exponentially with $\epsilon(z_i^k;\delta)$ grows. Based on this thought, we call Eq. 4 logarithmic loss function since $\forall i, H(Y_{tr})$ is fixed and maximizing Eq. 4 is equivalent to minimizing $H(Y_{tr}|Z_{tr}^k)$ which is a logarithmal like function. The difference between the two loss functions lies in that exponential loss function proves to be a "cautious" method for the graph construction as it imposes a significant penalty when the number of incorrect vertices becomes larger. On the other hand, a logarithmic loss function tolerates a large number of incorrect vertices and thus retrieves more (probably incorrect) edges into the graph.

	Parkour		Wedding Ceremony	
Rank	log-loss	exp-loss	log-loss	exp-loss
1	windows	walk/running	dresses	adult
2	walk/running	building	asian people	person
3	body parts	suburban	adult	body parts
4	road	outdoor	talking	male person
5	streets	face	adult female	standing

Table 1: Top 5 features selected by log and exp loss function on the events "parkour" and "wedding ceremony".

As it turns out both methods are good at exploiting high-level features which are highly related to an event. For example, Tab. 1 lists the top 5 high-level features selected by the two methods in terms of their loss (in decreasing order) on our experimental high-level feature set. It can be seen that both methods capture essential high-level features distinguishing the event from others. However, there is a huge disparity between their generated graphs, see Fig. 2, where the correct edge which connects two positive or two negative samples is marked in grey whereas the incorrect edge in red. The graph generated by the log-loss function is much more dense and includes more incorrect edges. For instance, the density of the graph illustrated in Fig. 2(a) is approximately 31 times that of Fig. 2(b), whereas the number of incorrect edges in Fig. 2(a) is about twice of that in Fig. 2(b).

Being "cautious" increases precision by preventing the error spreading throughout the graph but it decreases recall because some potential edges may be missing. We always face such a tradeoff between precision and recall in practice and the answer may vary given different datasets and evaluation criteria. However, it should be noted that the collective classification works on graphs where the error between vertices can also be propagated to the neighbors of nodes and finally to all vertices within the connected component. Sometimes a tiny error cascades onward until is renders the whole estimation inaccurate. Therefore, in a general sense, being too "aggressive" may not be a prudent policy.

3.3 Collective Inference

The goal of collective learning is to estimate a posterior probability distribution on class labels conditional on the established graph G and local hypothesis h_l. The learning algorithm directly models the marginal distribution rather than the joint distribution on the label, since the discriminative approach generally requires less training data. Suppose $\mathcal{N}_i$ represents a set of immediate neighbors of the vertex v_i. Following the first-order Markov assumption, given h_l, the label probability of vertex v_i only involves its neighbors $\mathcal{N}_i$, which can be formulated by [10]:

$$P(y_i|G,h_l) = P(y_i|\mathcal{N}_i,h_l) = \frac{1}{Z}\prod_{v_j\in\mathcal{N}_i} P(y_j|\mathcal{N}_j,h_l)^{w_{ij}} \quad (6)$$

where Z is a normalizer and h_l can be regarded as a prior embedded in the marginal distribution. In practice, log probability is applied to avoid the underflow and for the rest of the paper, we assume the log probability is used.

According to Eq. 6, the label probability of a vertex is determined by the weighted combination of those of its neighbors. Because of the recursive definition, the probability of each vertex must be inferred simultaneously. In the rest of this subsection, we will discuss two famous algorithms and how they solve the inferencing problem.

Gibbs sampling [16] becomes increasingly popular recently as it is broadly regarded as the most accurate approximate inference algorithm [9]. The basic idea of Gibbs sampling is to iteratively sample the label probability over their neighbors, see Algorithm 2. Specifically, first, we initialize the posterior probability with the prior distribution h_l. For convenience, $y_i^{(j)}$ denotes $P(y_i|G,h_l)^{(j)}$ and let the superscript

[1]Relaxing the assumption is beyond the topic of this paper. Interested audiences are encouraged to refer [10] for a discussion.

Algorithm 2: FFCC using Gibbs sampling

input : a graph G, a local hypothesis h_l
output: a posterior distribution $P(y|G,h_l)$

1 **foreach** $v_i \in V$ **do** $y_i^{(0)} = h_l(v_i)$;
2 **for** $j = 1$ **to** 200 **do** //brun-in
3 Generate a ordering O over V and $Q \leftarrow \varnothing$;
5 **while** *O is not empty* **do**
6 $y_i^{(j)} = \sum_{v_k \in (\mathcal{N}_i \cap Q)} w_{ik} y_k^{(j)} + \sum_{v_k \in (\mathcal{N}_i \cap O)} w_{ik} y_k^{(j-1)}$;
7 $O = O \backslash v_i$; $Q = Q \cup v_i$;
8 **end**
9 normalize l ;
10 **end**
11 initialize a label vector $l^T = (y_1^{(200)}, ...y_N^{(200)})^T$;
12 **for** $j = 201$ **to** 2200 **do** //sampling
13 Generate a ordering O over V and let $Q \leftarrow \varnothing$;
14 **while** *O is not empty* **do**
15 $y_i^{(j)} = \sum_{v_k \in (\mathcal{N}_i \cap Q)} w_{ik} y_k^{(j)} + \sum_{v_k \in (\mathcal{N}_i \cap O)} w_{ik} y_k^{(j-1)}$;
16 $l_i = l_i + y_i^{(j)}$;
17 **if** $j \bmod T = 0$ **then**
18 **foreach** $v_k \in V$ **do** $l_k = \frac{l_k}{T+1}$;
19 **end**
20 $O = O \backslash v_i$; $Q = Q \cup v_i$;
21 **end**
22 normalize l ;
23 **end**
24 **return** l

indicates the iteration number in Gibbs sampling. Step 2 through 9 is a period called "burn-in", in which some iterations are ignored in order to eliminate some inaccurate estimations occurring at the beginning of the sampling. In Step 3, we create a random ordering over all vertices in G and a set Q to store all vertices that currently have been sampled. In Step 6, the label probability y_i is updated using the log form of Eq. 6. Those $y_k^{(j)}$ are applied on the right side of the equation if they have been sampled at iteration j, otherwise $y_k^{(j-1)}$ are applied instead. After the vertex v_i is sampled, sets O, Q are updated accordingly in Step 7. Step 11 through 20 is the period where we actually sample a posterior distribution from the given graph and prior. It consists of the same steps as those in burn-in period excepts that a row vector l is introduced to maintain the sampling statistics and is updated very T iterations (see Step 17). After statistics are collected through sufficient iterations, the algorithm outputs the normalized posterior probability represented by l. In practice, it is difficult to decide on a sufficient number of iterations for both burn-in and sampling period as the convergence tests are neither robust nor well understood [16]. In this paper, we follow the commonly predefined settings in [10, 9] and run 200 and 2000 iterations for burn-in and sampling period, respectively.

Although Gibbs sampling proves to be an effective collective classification approach over many datasets, there is no guarantee that Gibbs sampling will converge within an acceptable number of iterations. In this paper, we instead propose to use Markov random walk which is also a famous model but has not been studied as a collective classification algorithm. The basic idea can be interpreted as the following process: the posterior probability y_i of a vertex v_i is updated by a random walker, who jumps to any of v_i's neighbors with probability d and jumps back to the local hypothesis $h_l(v_i)$ with probability $1-d$. d is a constant number, called the damping factor, measuring the frequency of the jump back. The above process is a Markov chain because of the first-order Markov assumption. To model the above process, we modify Eq. 6 by introducing an additional term quantifying the probability updates of the jump back, which gives us:

$$P(y_i|\mathcal{N}_i, h_l) = \frac{1}{Z'}((1-d)h_l(v_i) + d \sum_{v_j \in \mathcal{N}_i} \frac{w_{ij} P(y_j|\mathcal{N}_j, h_l)}{\sum_{v_k \in \mathcal{N}_j} w_{jk}}) \quad (7)$$

where Z' is a normalizer and $h_l(v_i)$ represents the local hypothesis for vertex v_i. Note as d decreasing, Eq. 7 imposes more weights on the prior h_l and when $d = 0$, it degenerates to the local hypothesis, in which only low-level features are considered. In contrast, when $d = 1$ it ignores the prior and the process may be unstable and thus may not converge. The denominator of the second part counts the degree (precisely out-degree) of vertex v_j in the weighted graph G and for the isolated vertices we add a self-loop to avoid the denominator becoming 0. Compared with Eq. 6, the probability passing from vertex v_j to v_i is determined not only by $P(y_j|\mathcal{N}_j, h_l)$ and w_{ij} but also by v_j's out-degree and the more neighbors a node has, the less probability each of its neighbors will receive. In other words, the probability passing from a vertex is locally normalized by its out-degree.

Following the markov process, we define the Markov transition matrix $\mathbf{M}$ as:

$$\mathbf{M}_{ij} = (1-d)h_l(v_i) + \frac{d \times w_{ij}}{\sum_{v_k \in \mathcal{N}_j} w_{jk}}, \quad (8)$$

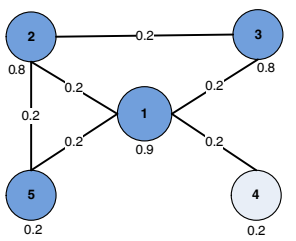
based on which the collective classification by Markov random walk can be brought forward, see Algorithm 3. Similar to Gibbs sampling, it initializes the posterior probability with the prior h_l. The posterior probability is then updated by Eq. 7 in Step 5 until it eventually converges. Following the graph construction method discussed in Section 3.2, the convergence is guaranteed(see the proof in Appendix). Furthermore based on our empirical observations, the algorithm converges very fast, typically in fewer than 20 iterations.

Algorithm 3: FFCC using Random Walk

input : a graph G, a local hypothesis h_l
output: a posterior distribution $P(y|G,h_l)$

1 **foreach** $v_i \in V$ **do** $y_i^{(0)} = h_l(v_i)$;
2 $j = 0$;
3 **do**
4 $j = j + 1$;
5 $\mathbf{y}^{(j)} = \mathbf{M}\mathbf{y}^{(j-1)}$
6 **while** $\|\mathbf{y}^{(j)} - \mathbf{y}^{(j-1)}\|_2 < \eta$;
7 normalize $\mathbf{y}$;
8 **return** $\mathbf{y}$

Compared with the Markov Random walk algorithm, the performance of Gibbs sampling may vary as its convergence is not guaranteed. We study the comparison on a tractable example that allows for clearer diagnosis, illustrated in Fig. 3. Suppose we have 4 positive samples: v_1,v_2,v_3 and v_5 and a negative sample v_4. The local hypothesis h_l is depicted under each vertex. Since high-level features are imperfect, the



Gibbs Sampling

	r1	r2	r3
1	0.968	0.729	1.000
2	0.748	0.735	0.633
3	0.504	0.419	0.339
4	0.291	0.364	0.493
5	0.393	0.654	0.399

Random Walk

	r1	r2	r3
1	0.957	0.957	0.957
2	0.736	0.736	0.736
3	0.532	0.532	0.532
4	0.232	0.232	0.232
5	0.442	0.442	0.442

Figure 3: Comparison between Gibbs sampling and Random Walk on a tractable example. Shaded vertices are positive samples. The number on edge indicates the weight and the number below each vertex is the probability outputted by local hypothesis. The two tables on the right are the updated probability under different runs

graph usually includes incorrect edges, e.g. (v_1, v_4). We use Non-interpolated Average Precision (AP) as the evaluation criteria. The AP for the local hypothesis is 0.95. We run Gibbs Sampling and Random Walk (d = 0.85) three times and list the result in the two tables in Fig. 3. Since the convergence in Random Walk is guaranteed, all three runs converge to the same distribution, where the AP is improved to 1.0 since the probability of v_5 is increased by receiving the probability passed from v_2 and v_3. On the other hand, Gibbs sampling yields 3 different results. Though the first run (column r1) resembles the Random Walk's estimation, the other two runs are quite different. Moreover, the third run's MAP is 0.888 even worse than the local hypothesis. Therefore it is worth noting that Gibbs Sampling may yield different results even with the same configuration. Random Walk is therefore a recommendable alternative because of its convergence property.

3.4 Complexity Analysis

This subsection discusses the computational complexity analysis for the proposed method, which may shed a light on the method's scalability as datasets grow in size. Recall that the dimension of high-level feature vector is M_h, low-level features M_l, and the number of total samples in the training set equals to N. Regarding the graph construction, since the two methods discussed in Sec. 3.2 are essentially the same method with different loss functions, they share the same upper bound $O(M_h(N + N^2)) = O(M_h N^2)$. The time complexity for Gibbs sampling is $O(M_h(B + S)N^2)$, where B and S are two constants representing the number of iterations for burn-in and sampling period, respectively. Random walk is $O(RM_h N^2)$, where R is the constant number representing the average iterations to converge. Based on the assumption that $B + S = 2200$ and $R \leq 20$, random walk is expected to be much more efficient than Gibbs Sampling. The time complexity of training a local hypothesis depends on the model and usually is done once beforehand, therefore we skip this part in the analysis. Regarding the feature wrapper in Step 3 Algorithm 1, since we adopt a forward wrapper using a hill climbing search, the complexity becomes $O(M_h^2 N)$. Note that here we precompute and maintain the local and relational hypothesis so that they can be directly applied in the wrapper. As a result, the time complexity of the proposed method is $O(M_h N^2 + M_h^2 N)$ for both Gibbs sampling and Markov random Walk.

4. EXPERIMENTS

In this section, we conduct a series of experiments validating the claim that the proposed method not only improves the performance but also the interpretability over the state-of-the-art method. Specifically, we first compare the proposed method with the baseline method in Section 4.2, then we study the impact of parameters on system performance. Finally, the performance of different graph construction and collective classification algorithms are compared in the last subsection.

4.1 Setup

Using the MED11 dataset, we conducted the experiment on the labeled samples development set in TRECVID 2011 contest, which consists of 15 predefined events and 2049 video clips. The details about the dataset are presented in Figure 4. The datset is adopted mainly for two reasons: there are not many public datasets aviablable on event detection and TRECVID MED11 is one of the most representative and challenging public datset. MED11 embraces various types of video clips for each event, e.g. skating, skiing, surfing and even finger skating in the event "attempting a board trick".

Color SIFT (CSIFT) is adopted as the low-level feature. Soft histogram assignment[17] is applied while constructing the code book, with the number of centroids equaling to 4096. Then spatial Bag of Words (BoWs) with a Pyramid Match Kernel[11, 12] is also used to further improve the classification result. The local hypothesis is trained using SVM with Chi-square kernel and multi-class classification is achieved using one-versus-all SVM. The semantic indexing object bank from the TRECVID 2011 contest, which consists of 346 visual concepts, is incorporated as the high-level feature[19]. Each visual concept is encoded as a real value ranging from 0 to 1. The final hypothesis is derived from the average of relational hypothesis and smoothed by the local hypothesis. Regarding the collective classification algorithm, we set the damping factor d (see Algorithm 3) to 0.85 and T (see Algorithm 2) to 10. The default loss function is the exp-loss function and the collective classification algorithm is the Markov random walk.

We compare the proposed method with four baseline methods using 5-fold cross-validation. In the first three baseline methods we train an SVM with Chi-square kernel where the difference in methods lies in how they combine the high-level and low-level features. We call the first method Early Fusion by Feature Concatenation(EF-FC), which simply concatenates the low-level features and high-level features in a new high dimensional feature space with which Chi-square kernel is then calculated; the second method Early Fusion by Kernel Fusion (EF-FC) averages their kernel matrixes and then training a model using the new kernel matrix; the third one is called Late Fusion (LF), which averages the classification results of high-level and low-level features; The last method follows the idea in [14], where we first project the two original feature spaces into two latent spaces with the maximum correlation by Canonical Correlation Analysis (CCA)[18] and then we late fuse the outputs of the SVM classifiers that are trained on the latent spaces. We call it

CCA and according to [6] it achieves the best result given a linear correlation between high-level and low-level features.

Figure 4: Overview of MED 11 dataset

The standard metric used in TRECVID MED Contest Normalized Detection Cost (NDC) function [19] is adopted as the evaluation metric. NDC is a weighted linear combination of the system event specific missed detection and false alarm probabilities. For each event E we have:

$$NDC(E) = \frac{0.999 \times P_{FA}(E) + 0.08 \times P_{MD}(E)}{0.08} \quad (9)$$

where $P_{MD}(E)$ equals the number of miss detection divides the number of positive clips for event E and $P_{FA}(E)$ equals the number of false alarms over the number of negative clips. Minimum NDC (Min NDC) is introduced to ignore the decision threshold, which measures the best prediction of a method. Min NDC equals to 0 for the perfect model where $P_{FA}(E) = P_{MD}(E) = 0$ and to 1 for the mute model that classifies every sample as the negative sample.

4.2 Comparison with Baseline Methods

To evaluate the performance of the proposed method, we compare it with the baseline method and list the result in Tab. 2. As can be seen, the performance is boosted by fusing the low-level features and high-level features in all methods except for CCA. On average, the proposed method outperforms other fusion methods and achieves the best score in 12 out of 15 events. According to the t-test, it is significantly better than other baseline methods at the 0.005 level(P-value ≤ 0.0006). The average performance of EF-FC, EF-KF and LF are virtually the same whereas CCA has the poorest performance which suggests a non-linear correlation between the feature spaces in this problem. Regarding the runtime, on a single core Intel Core i7 CPU@2.8GHz with 4GB memory, the experiment of Alg. 1 takes around 18 hours for Markov random walk and 66 hours for Gibbs sampling using R (http://www.R-project.org).

As mentioned above, one of the merit that the proposed method enjoys is the interpretability. We analyze the fusion result by inspecting the high-level features used in each event. Generally, for the events the reason FFCC is better due to it selects the distinguishing high-level features reasonably summarizing the current event and separating it from the others. E.g. for the event "feeding an animal" (ID=2), it selects the features such as "cows", "cetacean", "male person", "forest", "birds" and etc.; for the event "wedding ceremony" (ID=4), such as "room", "adult", "child", "legs" and etc. We also found that some features that seems meaningless for some event turns out useful in improving the performance e.g. concept "airplane" in the event "wedding ceremony". It may be due to either these feature's corresponding patterns rarely present in the event or the unknown pattern that the high-level feature's classifier captures is actually useful. On the other hand, for some events the proposed method is worse because it fails to find the robust features relevant to the event. We conclude two reasons account for the failure. First the object bank we used is biased for different types of events. It embraces many features for some events, e.g. "feeding an animal" (ID=2) and "landing a fish" (ID=3) whereas only a few for the others e.g. "working on a woodworking" (ID=5). Second high-level features are not equally accurate. Therefore some inaccurate features result in poor performance even if they are relevant to the event.

ID	LL	HL	EF-FC	EF-KF	LF	CCA	FFCC
1	0.576	0.490	0.492	0.508	0.497	0.633	**0.454**
2	0.872	0.786	0.810	0.828	0.792	0.875	**0.743**
3	0.588	0.526	0.476	0.498	0.502	0.629	**0.408**
4	0.430	0.360	0.353	0.331	0.339	0.422	**0.250**
5	0.722	0.719	0.682	0.658	**0.652**	0.803	0.688
6	0.673	0.553	0.575	0.568	0.539	0.562	**0.481**
7	0.790	0.700	0.635	0.657	0.683	0.749	**0.542**
8	0.455	0.396	0.378	**0.343**	0.391	0.457	0.355
9	0.547	0.389	0.376	0.407	0.415	0.562	**0.347**
10	0.807	0.701	0.680	0.707	0.675	0.562	**0.611**
11	0.682	0.787	0.772	0.746	0.697	0.803	**0.566**
12	0.624	0.488	0.483	0.524	**0.472**	0.655	0.493
13	0.598	0.450	0.438	0.443	0.458	0.542	**0.429**
14	0.544	0.556	0.459	0.443	0.495	0.544	**0.408**
15	0.724	0.653	0.646	0.683	0.690	0.721	**0.543**
AVG	0.642	0.570	0.550	0.556	0.553	0.647	**0.488**

Table 2: Performance comparison with baseline methods in terms of Min NDC. LL for low-level features, HL high-level features, EF-FC early fusion by feature concatenation, EF-KF early fusion by kernel fusion, LF late fusion and FFCC for the proposed method. The best method for each event is in bold.

FFCC also allows for interpreting fusion results for individual video clips. Fig. 5 illustrates an example where we select the two video clips with the greatest change during the collective classification in the event "wedding ceremony".

Since the result interpretation is beyond the topic of this paper, here we adopt a simple way to calculate the contribution for each high-level feature. Given a high-level feature, we use its proportion in the difference between the final hypothesis and local hypothesis to represents its contribution. In the first video clip (HVC390469) the score is 0.359 according to the CSIFT feature whereas because of the presence of visual concept "kitchen" and "anchorperson" which rarely present in the event "weeding ceremony"; the score is decreased by 0.076 and 0.062. In the second video clip (HVC631950), because of the presence of visual concepts "legs", "adult" and "child" that are commonly seen in the event; the score is increased by 0.049, 0.094 and 0.034. It is worth mentioning that as the features are imperfect, some interpretation may disagrees with human's prior knowledge. Nevertheless the experiment demonstrates FFCC's capability in interpreting the final result, which allows for appreciating not only which features update the score, but also how much the change is.

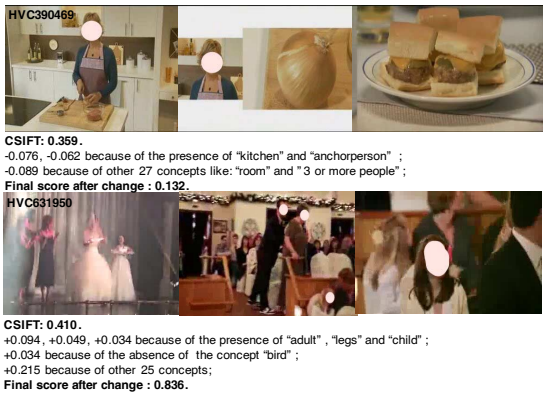
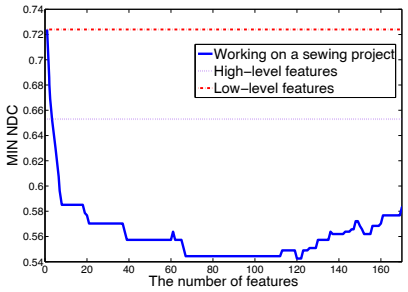


Figure 5: Interpreting the change for individual video clips

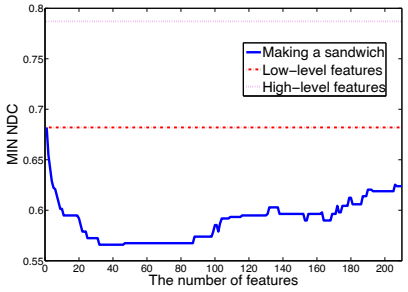
4.3 Parameter Changes

The number of high-level features incorporated in the final hypothesis is determined by the feature wrapper in Algorithm 1. However, an interesting question is that how does the number of features affect the final result? We conducted experiments to investigate how the performance changes when the number of high-level feature grows[2]. Fig. 6 illustrates the result on two representative events, in which x-axis represents the number of features incorporated and y-axis presents the corresponding Min NDC. At the origin no high-level feature is used, y value simply denotes the performance of low-level features. As it can be seen, at first the curve starts to fall sharply when the number of features grows and surpasses the baseline curves quickly. However, given a certain number of features have been incorporated, adding more features renders the curve rising. The underlying reason may be that due to the lack of sufficient information, less features often fails to estimate a good model so adding more features improves the performance. Nevertheless after sufficient relevant features are incorporated, adding more features may cause the overfitting thus the performance starts to drop. The subset of high-level features leads to the best Min NDC is called the optimal subset of features. Though the size of the optimal subset for all events is less than 125, the number for different events varies with the mean 68.7 and standard derivation 30.4. E.g. in Fig. 6(a) the size is 120 whereas in Fig. 6(b) the size equals to 32. Because the size of optimal subset is much less than the size of high-level features, which is 346, a considerable amount of resource could be saved when the method is applied to a large dataset.

[2]We did not reduce the feature dimension in the baseline methods as decreasing the number of high-level features in them leads to worse results.



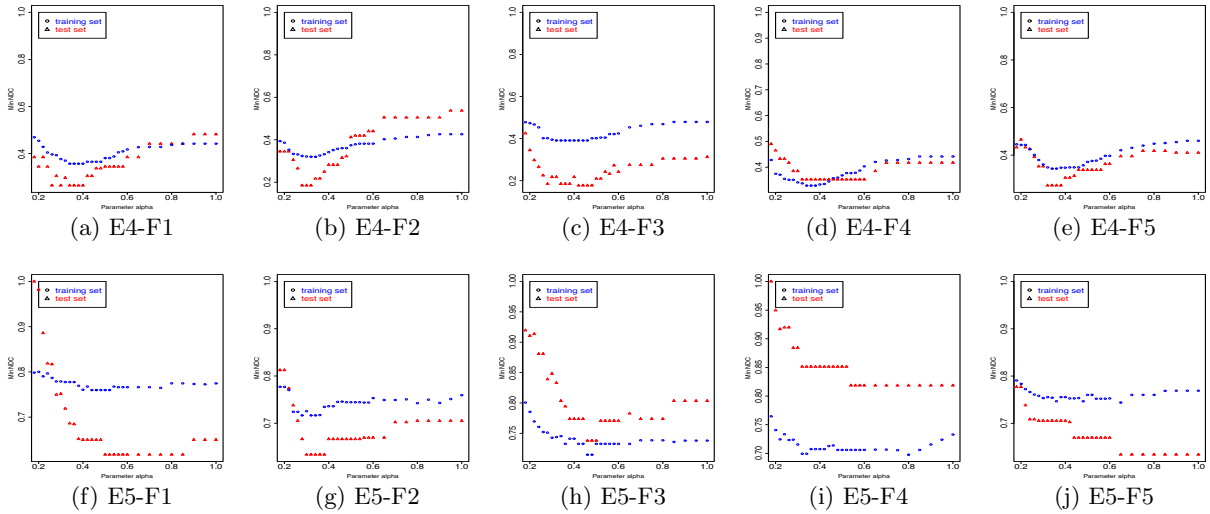
(a) working on a sewing project



(b) Making a sandwich

Figure 6: Performance change with the number of high-level features increased on two events.

Another observation we found is that unlike the other classification problem on social network, smoothing turns out to be very useful in our problem, see Eq. 2. The reason stems from the fact that number of training samples in our problem is insufficient to obtain an accurate final hypothesis, e.g. the feature dimensionality is 32768 but the number of positive training samples is less than 200. Recall the parameter α is leant in the training set and apply in the test set. We conducted the following experiments to study the impact of changes of α on the final performance. We analyze the event "wedding ceremony" (event ID 4) and "working on a woodworking" (event ID 5) where the proposed method has the greatest and least improvement over the baseline method. Fig. 7 presents the results in 5-fold in both events. The x-axis denotes the value of α and y-axis Min NDC value. The purple circle red triangle represents the Min NDC in training set and test set, respectively; the solid ones denotes the optimal α. Generally, the training and testing set follow the similar distribution and the smoothed hypothesis outperforms both the local and relational hypothesis while α scaling between 0.2 to 0.6.



(a) E4-F1 (b) E4-F2 (c) E4-F3 (d) E4-F4 (e) E4-F5

(f) E5-F1 (g) E5-F2 (h) E5-F3 (i) E5-F4 (j) E5-F5

Figure 7: The impact of parameter α in the 5 folds of the event "wedding ceremony"(E4) and "working on a woodworking"(E5)

4.4 Comparison among proposed methods

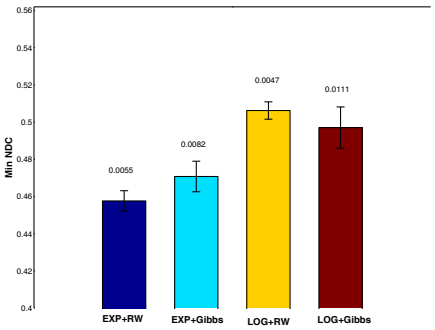


Figure 8: Comparison with different loss functions and collective classification algorithms. RW denotes for Random Walk , Gibbs for Gibbs Sampling, EXP for exponential loss function and LOG for logarithmic function. The number above each bar indicates the standard derivation.

In Section 3, we investigate two approaches to construct graphs by different loss functions and two collective classification algorithms namely Gibbs sampling and Markov random walk to inference posterior distributions. To evaluate the performance of different methods, we run each experiment 5 times with different cross validation partitions and report the mean and variance of the average minimum NDC, illustrated in Fig. 8 in which y-axis denotes for the average minimum NDC over 15 events. First of all, as can be seen, random walk with exponential loss function is the best configuration. In addition it enjoy a relative small variance. Generally, the variance of random walk is smaller than that of Gibbs Sampling because of its convergence property. Besides, the exponential loss function seems to be better than the logarithmic function, which suggests introducing too much error in graphs is harmful for the system.

5. CONCLUSIONS

In this paper, we approached Multimedia Event Detection with a novel method that fuses high-level and low-level features based on collective classification. Our method consists of three steps: training a classifier from low-level features; encoding high-level features into graphs; and diffusing the scores on the established graph to obtain the final prediction. We investigated two graph construction methods using logarithmic and exponential loss functions, respectively and two collective classification algorithms namely Gibbs sampling and Markov random walk. The theoretical analysis showed that the proposed Markov Random Walk converges and is computationally scalable, and thus superior to Gibbs sampling. The empirical analysis on the TRECVID 2011 Multimedia Event Detection dataset validates the method with very good performance compared to state-of-the-art methods, with an added benefit of interpretability. Besides, the comparison among the proposed method suggests that Markov random walk on graphs constructed by exponential loss function is the best configuration in our framework with respect to the minimum NDC.

6. ACKNOWLEDGMENTS

This work has been supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center contract number D11-PC20066. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DOI/NBC, or the U.S. Government.

7. REFERENCES

[1] I. Laptev, T. Lindeberg. Space-time interest points. In *ICCV*, pages 432–439, Nice, France, 2003.

[2] Li-Jia Li, Hao Su, Eric Xing, Fei-Fei Li. Object bank: a high-level image representation for scene

classification and semantic feature sparsification. In *NIPS*, pages 1378–1386, Vancouver, Canada, 2010.

[3] C. Snoek, M. Worring, A. W. M. Smeulders. Early versus late fusion in semantic video analysis. In *ACM Multimedia*, pages 399–402, Singapore, 2005.

[4] T. Pham, N. Maillot, J. Lim, J. Chevallet. Latent semantic fusion model for image retrieval and annotation. In *CIKM*, pages 439–444, Lisbon, Portugal, 2007.

[5] H. Escalante, C. Hernández, L. Sucar, M. Montes. Late fusion of heterogeneous methods for multimedia image retrieval. In *ACM MIR*, pages 172–179, Vancouver, Canada, 2008.

[6] J. Kludas, E. Bruno, S. Marchand-Maillet. Information fusion in multimedia information retrieval. In *Adaptive Multimedia Retrieval*, pages 147–159, Paris, France, 2007.

[7] L. Bao et al. Informedia@TRECVID 2011. In *Trecvid Video Retrieval Evaluation Workshop, NIST*, Gaitherburg, USA, 2011.

[8] H. Eldardiry, J. Neville. Across-Model collective ensemble classification. In *AAAI*, to appear, San Francisco, USA, 2011.

[9] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[10] S. Macskassy, and F. Provost. Classification in networked data: A toolkit and a univariate case study. *JMLR*, 8:935–983, 2007.

[11] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, pages 2169–2178, New York, USA, 2006.

[12] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *CIVR*, pages 401–408, Amsterdam, Netherlands, 2007.

[13] Y. Wu, E. Y. Chang, K. C. Chang, J. R. Smith. Optimal multimodal fusion for multimedia data analysis. In *ACM Multimedia*, pages 572–579, New York, USA, 2004.

[14] N. Rasiwasia, JC. Pereira, E. Coviello, G. Doyle, G. Lanckriet, R. Levy, N. Vasconcelos. A new approach to cross-modal multimedia retrieval. In *ACM Multimedia*, pages 251–260, Firenze, Italy, 2010.

[15] L. K. McDowell, K.M. Gupta, D.W. Aha. Cautious inference in collective classification. In *AAAI*, pages 596–601, Vancouver, Canada, 2007.

[16] W. R. Gilks,S. Richardson and D. J. Spiegelhalter. Markov chain Monte Carlo in Practice. *Chapman Hall/CRC Interdisciplinary Statistics*, 1996.

[17] J. Gemert, J. Geusebroek, C. Veenman, A. Smeulders. Kernel codebooks for scene categorization. In *ECCV*, pages 696–709, Marseille, France, 2008.

[18] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28:321–377, 1936.

[19] P. Over, G. Awad, J. Fiscus, B. Antonishek, and M. Michel. Trecvid 2010 - an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *Trecvid Video Retrieval Evaluation Workshop, NIST*, Gaitherburg, USA, 2010.

[20] Doeblin, W. Exposé sur la théorie des chaînes simples constantes de Markoff à un nombre fini d'états. *Rev. Math. Union Interbalkanique*, 2:77–105, 1938.

APPENDIX

THEOREM 1. *Let h_l be the local hypothesis, $h_{r_i} = P(y|G_i, h_l)$ be the i^{th} relational hypothesis and $h_f = P(y|G_1, ...G_n, h_l)$ be the final hypothesis. Assume $G_1, ...G_n$ are independent of each other then*

$$\log h_f = \sum_{i=1}^{n} \log h_{r_i} - (n-1) \log h_l$$

PROOF. For graphs $\{G_1, ...G_n\}$, according to Bayes rule we have:

$$P(y|G_1, ...G_n, h_l) = \frac{P(G_1, ...G_n|y, h_l)P(y|h_l)}{P(G_1, ...G_n|h_l)}.$$

Since $G_1, ...G_n$ are independent we have:

$$\begin{aligned} P(y|G_1, ...G_n, h_l) &= \frac{P(y|h_l)\prod_{i=1}^{n} P(G_i|y, h_l)}{\prod_{i=1}^{n} P(G_i)} \\ &= \frac{P(y|h_l)\prod_{i=1}^{n} P(G_i|y, h_l)}{\prod_{i=1}^{n} P(G_i)} \frac{P(y|h_l)^{n-1}}{P(y|h_l)^{n-1}} \\ &= \frac{1}{P(y|h_l)^{n-1}} \prod_{i=1}^{n} P(y|G_i, h_l) \end{aligned}$$

Taking the log on both side of the above equation:

$$\log P(y|G_1, ...G_n, h_l) = \sum_{i=1}^{n} \log P(y|G_i, h_l) - (n-1) \log P(y|h_l).$$

Since $P(y|h_l) = h_l$, substituting h_f and h_{r_i} for $P(y|G_1, ...G_n, h_l)$ and $P(y|G_i, h_l)$ gives the logarithmic form of the final hypothesis. □

THEOREM 2. *Algorithm 3 converges to a stationary distribution in finite time on the graph generated by Eq. 3.*

PROOF. Let $\pi_t(i) = P(y_i|G, h_l)$ denotes the label distribution over all $v_i \in V$ at time t and $\pi_0 = h_l$. The goal is to prove $\pi_t = \mathbf{M}\pi_{t-1} = \pi_{t-1}$ for some finite time t.

For each connect component C_j in the Graph G:

- if C_j contains a single vertex $v_i \in C_j$: for $t \geq 2$, we have $\pi_t(i) = \pi_{t-1}(i)$.
- if C_j contains more than one vertices: according to Eq. 3, $\forall v_i, v_j \in C_j$, there is an edge between from v_i to v_j. In other words, the subgraph C_j is strongly connected and any two vertices are mutually reachable. Therefore $\mathbf{M}$ is irreducible.

  Besides, for any pair of vertices in C_j as they are mutually reachable they must have the same period. Consequently all in C_j states have the same period. Furthermore, according to Eq. 7, each vertex has a self-loop edge therefore each vertex has the period 1 and thus $\mathbf{M}$ is aperiodic.

  As $\mathbf{M}$ is irreducible and aperiodic, then $\forall v_i \in C_j, \exists t$, s.t. $\pi_t(i) = \pi_{t-1}(i)$, i.e. the algorithm converges to a unique stationary distribution for all vertices in C_j[20].

Consequently, suppose the convergence time for connect component C_j is t_j. Let $t = \max_{C_j \subseteq G}(t_j)$, then $\forall v_i \in G$, $\pi_t(i) = \pi_{t-1}(i)$. In other words the label distribution converges to a stationary distribution $\pi_t(i)$ after t iterations. □