PROPOSTA E AVALIAÇÃO DE MECANISMOS DE COMBATE À POLUIÇÃO EM SISTEMAS DE COMPARTILHAMENTO DE VÍDEOS

HENDRICKSON REITER LANGBEHN

PROPOSTA E AVALIAÇÃO DE MECANISMOS DE COMBATE À POLUIÇÃO EM SISTEMAS DE COMPARTILHAMENTO DE VÍDEOS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Marcos André Gonçalves
Coorientador: Jussara Marques de Almeida

Belo Horizonte

Março de 2011

HENDRICKSON REITER LANGBEHN

PROPOSAL AND EVALUATION OF MECHANISMS TO COMBAT POLLUTION IN ONLINE VIDEO SHARING SYSTEMS

Dissertation presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Marcos André Gonçalves
Co-Advisor: Jussara Marques de Almeida

Belo Horizonte
March 2011

Langbehn, Hendrickson Reiter.

L271p Proposta e Avaliação de Mecanismos de Combate à Poluição em Sistemas de Compartilhamento de Vídeos / Hendrickson Reiter Langbehn. — Belo Horizonte, 2011.
xxxii, 55 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais. Departamento de Ciência da Computação.

Orientador: Marcos André Gonçalves.
Coorientadora: Jussara Marques de Almeida.

1. Computação - Teses. 2. Visão por computador - Teses. I. Orientador. II. Coorientadora. III. Título.

CDU 519.6*84 (043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Proposta e avaliação de mecanismos de combate à poluição em sistemas de compartilhamento de vídeos

HENDRICKSON REITER LANGBEHN

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. MARCOS ANDRÉ GONÇALVES - Orientador
Departamento de Ciência da Computação - UFMG

PROF. ALTIGRAN SOARES DA SILVA
Departamento de Ciência da Computação - UFAM

PROF. ADRIANO ALONSO VELOSO
Departamento de Ciência da Computação - UFMG

PROFA. JUSSARA MARQUES DE ALMEIDA
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 28 de março de 2011.

"Freedom is not worth having if it does not
connote freedom to err and even to sin.
If God Almighty has given the humblest
of His creatures the freedom to err,
it passes my comprehension how human beings,
be they ever so experienced and able,
can delight in depriving other human beings
of that precious right."
(Mahatma Gandhi)

Resumo

A maioria dos sistemas de compartilhamento de vídeo online (SCVOs), como o YouTube e o Yahoo! Vídeo, possuem vários mecanismos para suportar interações entre os usuários. Um destes mecanismos é o recurso de vídeo-resposta no YouTube, que permite ao usuário postar um vídeo em resposta a um outro vídeo. Embora cada vez mais popular, o recurso de vídeo-resposta abre a oportunidade para que usuários não-cooperativos introduzam "conteúdo poluído" no sistema, causando perda de eficácia e credibilidade do serviço, bem como desperdício de recursos do sistema. Por exemplo, os usuários não-cooperativos, a quem nos referimos como *spammers*, podem postar vídeos não relacionados em resposta a um outro vídeo (o vídeo respondido), tipicamente um vídeo muito popular, com o objetivo de ganhar visibilidade para seus próprios vídeos. Além disso, os usuários referidos como *promotores de conteúdo* postam diversos vídeos não relacionados em resposta a um único vídeo com a intenção de aumentar a visibilidade deste último.

Trabalhos anteriores sobre a detecção de *spammers* e promotores de conteúdo no YouTube se basearam principalmente em métodos de classificação supervisionados. A desvantagem da aplicação de soluções supervisionadas para esse problema específico é que, além de extremamente caras (em alguns casos, milhares de vídeos tem que ser vistos e rotulados), o processo de aprendizagem tem de ser continuamente realizado para lidar com as mudanças nas estratégias adotadas pelos usuários não-cooperativos. Neste trabalho, exploramos o uso de estratégias semi-supervisionadas baseadas em múltiplas visões, o que nos permite reduzir significativamente a quantidade de treinamento para detectar usuários não-cooperativos no YouTube, mas mantendo uma eficácia similar àquela obtida utilizando todo o treinamento. Nosso método proposto explora o fato de que, neste problema, existe uma partição natural do espaço de atributos em sub-grupos ou "visões", cada uma sendo capaz de classificar usuários, quando dados de treino suficientes estão disponíveis. Além disso, propomos lidar com o problema da combinação de visões como um problema de agregação de rankings, onde rankings baseados na confiança da classificação são combinados para decidir se um exemplo não rotulado

deve ser incluído no conjunto de treino. Nossos resultados demonstram que somos capazes de reduzir a quantidade de treino em cerca de 80%, sem perdas significativas na efetividade da classificação.

Por fim, desenvolvemos um modelo analítico para estimar os custos associados com a utilização de diferentes métodos para identificar usuários não-cooperativos em SCVOs. Aplicamos este modelo em diversos cenários com o intuito de comparar nosso melhor método proposto (um método híbrido) com um método supervisionado que utiliza todo o conjunto de treino disponível (nosso *baseline*). Os resultados desta análise mostraram que nosso método possui um custo menor de utilização do que o *baseline* para grande parte dos cenários analisados.

Abstract

Most online video sharing systems (OVSSs), such as YouTube and Yahoo! Video, have several mechanisms for supporting interactions among users. One such mechanism is the video-response feature in YouTube, which allows a user to post a video in response to another video. While increasingly popular, the video-response feature opens the opportunity for non-cooperative users to introduce "content pollution" into the system, thus causing loss of service effectiveness and credibility as well as waste of system resources. For instance, non-cooperative users, to whom we refer as *spammers*, may post unrelated videos in response to another video (the responded video), typically a very popular one, aiming at gaining visibility towards their own videos. In addition, users referred to as *content promoters* post several unrelated videos in response to a single responded one with the intent of increasing the visibility of the latter.

Previous work on detecting spammers and content promoters on YouTube has relied mostly on supervised classification methods. The drawback of applying supervised solutions to this specific problem is that, besides extremely costly (in some cases thousands of videos have to be watched and labeled), the learning process has to be continuously performed to cope with changes in the strategies adopted by non-cooperative users. In this work, we explore the use of multi-view semi-supervised strategies, which allows us to reduce significantly the amount of training, to detect non-cooperative users on YouTube, while keeping high levels of effectiveness. Our proposed method explores the fact that, in this problem, there is a natural partition of the feature space in subgroups or "views", each being able to classify a given user when enough training data is available. Moreover, we propose to deal with the problem of view combination as a rank aggregation problem, where rankings based on confidence in the classification are combined to decide whether an unlabeled example should be included in the training set. Our results demonstrate that we are able to reduce the amount of training in about 80% without significant losses in classification effectiveness.

Finally, we develop an analytical model to estimate the costs associated with the utilization of different methods to identify non-cooperative users in OVSSs. We here

apply this model in different scenarios in order to compare our best proposed method (a hybrid method) with a supervised method which uses all the training data available (our baseline). The results of this analysis showed that our method has a lower cost when compared to the baseline for most of the analyzed scenarios.

Resumo Estendido

Introdução

Com a popularização da Web no últimos anos, o número de pessoas que usam a Internet cresce cada vez mais e uma parcela significativa desses usuários assistem vídeos online. Devido a essa demanda, sistemas de compartilhamento de vídeos online (SCVOs), tais como o YouTube e Yahoo! Video estão tendo um crescimento vertiginoso de popularidade. O fato de os usuários poderem criar seus próprios vídeos e postá-los na Web contribui muito para a popularidade dos SCVOs. Normalmente, SCVOs tem vários mecanismos para facilitar a recuperação dos vídeos. Um exemplo é a possibilidade de um usuário responder a um vídeo publicando um outro vídeo em resposta a ele, como no recurso de vídeo-resposta fornecido pelo YouTube. Um usuário é capaz de ver todas as respostas postadas a um determinado vídeo, que supostamente também conteria conteúdo de interesse, sem ter que fazer uma nova pesquisa.

Os fatores mencionados acima abrem espaço para ações não-cooperativas por parte dos próprios usuários. Benevenuto et al. [2009b] encontrou evidências de usuários não-cooperativos explorando o recurso de vídeo-resposta no YouTube. Esses usuários postam vídeos completamente não relacionados em resposta a outros vídeos. De acordo com Benevenuto et al. [2009a], estes usuários não-cooperativos podem ser classificados em dois tipos: *spammers* e *promotores de conteúdo*. *Spammers* são os usuários que postam vídeos não relacionados em resposta a vídeos populares com o intuito de aumentar a visibilidade de seus próprios vídeos. Um promotor é um usuário que tenta dar visibilidade para um vídeo respondido, postando um grande número de vídeos, na sua maioria não relacionados, em resposta a ele.

Conteúdo poluído traz várias desvantagens para os SCVOs, incluindo: (1) perda de efetividade e credibilidade do serviço, (2) desperdício de espaço, (3) desperdício de banda, e (4) perda de efetividade de caches e redes de distribuição de conteúdo.

O primeiro e, pelo que sabemos, único esforço para abordar o problema da poluição de conteúdo em SCVOs foi realizado em Benevenuto et al. [2009a]. Os

autores propõem mecanismos baseados em classificação para identificar os usuários que são *spammers* e promotores, diferenciando-os de usuários legítimos do sistema. Aplicando um algoritmo de classificação supervisionada em uma coleção de 829 usuários pré-classificados, os autores foram capazes de detectar a grande maioria dos promotores, bem como uma parcela significativa dos *spammers*.

Métodos supervisionados precisam "aprender" uma função de classificação através de um conjunto de dados de treino. A desvantagem da aplicação de tais métodos para detectar usuários não-cooperativos em SCVOs é que a geração manual da base de treino é muito custosa, já que milhares de vídeos devem ser analisados[1]. Um melhor compromisso entre o custo e a efetividade da classificação pode ser conseguido com métodos semi-supervisionados, os quais combinam uma quantidade menor de dados rotulados com uma grande quantidade de dados não rotulados para melhorar a classificação.

Neste trabalho, exploramos estratégias de classificação semi-supervisionada com múltiplas visões, o que nos permite reduzir significativamente a quantidade de treino necessário para identificar *spammers* e promotores em SCVOs, enquanto mantendo níveis de efetividade similares àqueles obtidos quando usando todo o treinamento. Nossos métodos propostos exploram o fato de que, neste problema, há uma partição natural do espaço de atributos em sub-grupos ou "visões", cada uma sendo capaz de classificar um determinado usuário quando uma quantidade suficiente de dados de treino está disponível. Assim, é possível combinar as visões para permitir que dados não rotulados sejam usados para aumentar um conjunto muito menor de dados rotulados.

Além disso, desenvolvemos um modelo analítico para estimar os custos associados com a utilização de diferentes métodos para identificar usuários não-cooperativos em SCVOs. Aplicamos este modelo em diversos cenários com o intuito de comparar nosso melhor método proposto (um método híbrido) com um método supervisionado que utiliza todo o conjunto de treino disponível (nosso *baseline*). Os resultados desta análise mostraram que nosso método possui um menor custo de utilização do que o *baseline* para grande parte dos cenários analisados.

Abordagem Semi-supervisionada com Múltiplas Visões

A principal idéia por trás de nossa proposta de classificação semi-supervisionada com múltiplas visões é começar com um pequeno conjunto de dados rotulados como sendo o conjunto de treino e ser capaz de expandi-lo com exemplos extraídos de um conjunto

[1]Em Benevenuto et al. [2009a] mais de 20.000 vídeos foram analisados manualmente.

de dados não rotulados. O conjunto de treino expandido é então utilizado para classificar os objetos desejados. Durante a inserção de novos elementos ao conjunto de treino, este e o conjunto de dados não rotulados são divididos em visões, cada uma sendo composta de uma série de atributos dos elementos. Quando não há mais elementos que possam ser inseridos no conjunto de treino, todas as visões do conjunto de treino formado são integradas em uma única, a qual é utilizada como o conjunto de treino final usado na classificação.

Algoritmo 1 Expansão do Treino Utilizando o Método Semi-supervisionado com Múltiplas Visões

Input: O número de visões V, um conjunto C de classificadores C_v para cada visão v ($v = 1...V$), o número de classes K, o número de elementos M no conjunto de dados não rotulados, um conjunto L de dados rotulados l_v^n com os atributos de cada visão v ($v = 1...V$) para cada elemento l^n ($n = 1...N$), e um conjunto U de dados não rotulados u_v^m com os atributos de cada visão v ($v = 1...V$) para cada elemento u^m ($m = 1...M$). {N é o número de elementos no conjunto de dados rotulados}

Output: Conjunto L expandido para incluir todos os elementos iniciais bem como os novos elementos adicionados pelo algoritmo.

1: **repeat**
2: $insertion \leftarrow FALSE$
3: **for** $k \leftarrow 1$ to K **do**
4: $B_k \leftarrow$ a fração de elementos da classe k em L
5: **end for**
6: **for** $v \leftarrow 1$ to V **do**
7: Treina C_v em L_v
8: Avalia C_v em U_v gerando as predições em P_v e as confianças em θ_v {θ é um conjunto de confianças $\theta_v^m(k)$ de cada visão v da pertinência do elemento u^m à classe k}
9: **end for**
10: {I recebe os elementos a serem adicionados ao conjunto de treino juntamente com a classe predita para cada elemento}
11: $I \leftarrow$ DefineNovasInserções(V, K, B, M, UP, θ)
12: **if** $|I| > 0$ **then**
13: $insertion \leftarrow TRUE$
14: **end if**
15: **for** $i \leftarrow 1$ até $|I|$ **do**
16: **for** $v \leftarrow 1$ até V **do**
17: $L_v \leftarrow L_v \cup u_v^i \in I$
18: $U_v \leftarrow U_v \setminus u_v^i \in I$
19: **end for**
20: **end for**
21: $M \leftarrow |U|$ {Atualiza o número de elementos no conjunto de dados não rotulados}
22: **until** ($|U| = 0$ ou $insertion = FALSE$)

Esta inserção de exemplos no conjunto de treino ocorre através de um processo iterativo, como mostrado no Algoritmo 1. Em cada iteração, o classificador de cada visão é treinado com os seus respectivos conjuntos de treino, e é utilizado para predizer a classe de cada elemento do conjunto de dados não rotulados. Juntamente com as predições, a etapa de avaliação gera também a confiança de cada elemento em pertencer à classe predita. Com base nas predições e na confiança da predição de cada classificador, um método é usado para determinar se existem elementos não rotulados que podem ser inseridos no conjunto de treino e para definir a classe predita de cada um desses elementos (função DefineNovasInserções). Em seguida, os elementos

selecionados são movidos do conjunto de dados não rotulados para o conjunto de treino e têm a sua classe atualizada para a classe predita. Este processo é repetido até que não existam mais elementos que possam ser inseridos no conjunto de treino ou até que o conjunto de dados não rotulados esteja vazio.

Usamos o Lazy Associative Classification (LAC) como nosso classificador (Veloso et al. [2006]). O LAC explora o fato de que, frequentemente, há fortes associações entre valores de atributos e classes. Tais associações estão normalmente escondidas nos dados de treino e, quando descobertas, podem revelar aspectos importantes que podem ser usados para prever classes de elementos. O LAC produz uma função de classificação composta por regras $X \rightarrow k$, indicando a associação entre um conjunto de valores de atributo X e uma classe k.

Combinação dos Resultados das Múltiplas Visões

Um passo importante da abordagem semi-supervisionada com múltiplas visões é a seleção dos elementos a serem inseridos no conjunto de treino em cada iteração, o que é feito através da combinação dos resultados de classificação de cada visão (função DefineNovasInserções no Algoritmo 1). Neste trabalho, exploramos duas estratégias para isso: uma baseada na concordância das visões e outra baseada em um método de agregação de listas ordenadas chamado Borda Count (Black [1963]).

Durante nossos experimentos iniciais, descobrimos que é muito importante, por razões de efetividade da classificação, manter estável a distribuição de elementos rotulados por classe, à medida que novos elementos são inseridos no conjunto de treino. Portanto, nós calculamos a distribuição inicial de elementos entre as classes no conjunto de treino e a usamos para restringir a inserção de novos elementos em cada iteração. Esta distribuição está representada pelo conjunto B no Algoritmo 1.

A estratégia de concordância de visões funciona como explicado a seguir. Todos os elementos do conjunto de dados não rotulados são analisados. Se todas as visões concordam na predição da classe de um elemento, este se torna um candidato a ser adicionado no conjunto de treino. Após o término do processo de seleção de candidatos, usamos a confiança das predições e a distribuição de elementos para a classe em questão para determinar quais elementos do conjunto de candidatos serão inseridos no conjunto de treino.

A estratégia com o Borda Count é baseado em um método que foi proposto para combinar listas de candidatos em eleições e, depois, foi utilizado para resolver problemas computacionais como a agregação dos rankings produzidos por diversas máquinas de busca (Dwork et al. [2001]). Esta estratégia calcula valores de ranking

proporcionais à posição de cada elemento, para todos os elementos do conjunto de dados não rotulados, em relação a uma classe. Na fase de agregação, os valores individuais gerados por cada visão com relação a esta mesma classe são somados. Os elementos com os valores mais elevados de agregação de ranking, que tiveram sua classe predita como a classe em questão por pelo menos uma visão, são selecionados para serem inseridos no conjunto de treinamento, de forma que a distribuição dos elementos entre as classes não seja alterada.

Observe que ambos os métodos fazem o cálculo do conjunto de elementos selecionados para todas as classes. No entanto, o conjunto de elementos candidatos é determinado independentemente para cada classe. Assim, diferentes estratégias podem ser aplicadas a cada classe, dependendo das características do problema a ser resolvido. De fato, nós exploramos abordagens híbridas (i.e., combinando estratégias) em nossos experimentos descritos mais adiante, como veremos.

Metodologia de Avaliação

Nesta seção, descreveremos a coleção utilizada, as métricas de avaliação das soluções e, por fim, o ambiente experimental.

Coleção de Testes

Para avaliar as abordagens propostas precisamos de uma coleção de testes composta por usuários do sistema alvo, que no nosso caso é o YouTube. O processo de criação desta coleção é muito custosa, uma vez que requer o esforço humano para assistir a um número potencialmente muito grande de vídeos. Neste trabalho, utilizamos a mesma coleção de testes descrita em Benevenuto et al. [2009a], que tem um total de 829 usuários, constituída de 641 usuários legítimos, 157 *spammers* e 31 promotores.

Usuários legítimos, *spammers* e promotores têm objetivos diferentes no sistema e, portanto, espera-se que atuem de maneira diferente enquanto o utilizam. Tais diferenças podem ser capturadas explorando atributos que reflitam o comportamento de cada usuário quando utilizando o sistema. Em particular, a nossa coleção de testes contém um total de 60 atributos por usuário, que podem ser divididos em três grupos: atributos de vídeo, atributos de usuário e atributos da rede social estabelecida entre os usuários através da utilização do recurso de vídeo-resposta.

Diferentemente do que é feito em Benevenuto et al. [2009a], onde os autores aplicaram um método de classificação supervisionada com uma única visão, nosso objetivo aqui é explorar abordagens semi-supervisionadas com múltipals visões.

Assim, precisamos extrair diferentes visões da coleção de testes. Como explicado anteriormente, a coleção já possui três grupos distintos de atributos, ou seja, atributos do usuário, atributos de vídeo e atributos de rede social. Assim, aproveitamos esta categorização inerente dos atributos na coleção para gerar uma visão de vídeo, uma visão de usuário e uma visão de rede social. Cada visão inclui apenas os atributos do grupo correspondente. Em outras palavras, a visão de vídeo tem 42 atributos, a visão de usuário tem 10 atributos, e a visão de rede social tem 8 atributos.

Métricas Utilizadas

Nós avaliamos as abordagens de classificação comparando as matrizes de confusão (Kohavi and Provost [1998]) produzidas por cada uma delas. Cada elemento da posição (i, j) dessa matriz representa o percentual de usuários de classe i que foram preditas, pela classificação, como pertencentes à classe j. Nós focamos nossa avaliação nessas matrizes porque elas mostram melhor os compromissos entre classificar corretamente os usuários de uma classe em detrimento de classificar erroneamente usuários de outras classes.

Ambiente Experimental

Foi realizada uma série de experimentos com cinco abordagens de classificação. Três abordagens são baseadas no método semi-supervisionado com múltiplas visões, e exploram as duas estratégias de combinação de visões apresentadas. Para avaliar a efetividade dessas abordagens, também consideramos dois *baselines*. O primeiro é um método supervisionado com uma única visão que utiliza todos os dados de treino disponíveis. A comparação com este *baseline* permite que avaliemos o compromisso entre a quantidade de dados rotulados e a eficácia da classificação. Como um segundo *baseline*, consideramos o mesmo método supervisionado com uma única visão mas que use a mesma quantidade de dados rotulados que as estratégias propostas. A comparação com este segundo *baseline* nos permite avaliar o impacto sobre a classificação quando incorporamos novos exemplos para o conjunto de treino.

Os experimentos de classificação foram realizados utilizando uma validação cruzada de 5-folds e foram repetidos cinco vezes, usando diferentes sementes para embaralhar o conjunto de dados original. Assim, os resultados apresentados para cada uma das abordagens consideradas são médias das 25 execuções.

Resultados

Esta seção apresenta o resultado mais relevante da nossa comparação das diferentes abordagens de classificação consideradas. Este melhor resultado foi obtido com a nossa abordagem híbrida (explicada mais adiante).

O objetivo aqui é conseguir o melhor compromisso entre a quantidade de dados de treino e a efetividade de classificação da abordagem. Para avaliar esse compromisso, fizemos um conjunto de experimentos iniciais com nossas estratégias propostas, reduzindo cada vez mais a porcentagem dos dados de treino original fornecidos como dados rotulados, deixando o restante como dados não rotulados. Testamos várias percentagens, e descobrimos que usando apenas 20% dos dados de treino original levou ao melhor compromisso. Assim, focamos a nossa comparação nos resultados quando 20% do conjunto de treino original é usado como dados rotulados pelas abordagens semi-supervisionadas.

Começamos nossas análises, considerando o desempenho do nosso *baseline* treinado com todos os dados de treino disponíveis (Tabela 1). Como podemos ver, promotores e usuários legítimos são classificados corretamente em quase 100% dos casos, mas apenas 53% dos *spammers* estão corretamente classificados. Uma investigação revelou ainda que vários desses *spammers* são realmente muito difíceis de identificar com base apenas em seus comportamentos, pois eles possuem um comportamento dual, i.e., ora agem como legítimos e ora como *spammers*.

		Predito		
		Promotor	Spammer	Legítimo
Real	Promotor	100%	0%	0%
	Spammer	1.02%	53.25%	45.73%
	Legítimo	0%	0.78%	99.22%

Tabela 1: Classificação com o *Baseline* 1 (Método Supervisionado com 100% de Treino)

Claramente, *spammer* é a classe mais difícil de prever. Na abordagem de Concordância de Visões, existe uma concordância muito mais baixa entre as visões com relação a essa classe específica, o que pode afetar a classificação de todas as três classes, uma vez que provoca a interrupção do processo de inserção de novos dados (de todas as classes) ao conjunto de treino. Apesar disso, os resultados da Concordância de Visões para os promotores e usuários legítimos são razoavelmente boas. Assim, exploramos uma abordagem híbrida que aplica o algoritmo Borda Count apenas para os *spammers*, mantendo a Concordância de Visões para as outras 2 classes.

A Tabela 2 mostra a matriz de confusão com os resultados da abordagem híbrida.

Em comparação com o *baseline* com 100% de treino, esta abordagem é um pouco pior na predição de promotores e usuários legítimos. No entanto, alcança um desempenho comparável na identificação correta de *spammers*, a classe mais difícil. Estes resultados são bastante promissores considerando a grande redução (por um fator de 5) na quantidade de dados rotulados necessários. Além disso, se considerarmos a aplicação de nossa técnica como uma ferramenta para ajudar os administradores de sistema a filtrarem usuários suspeitos para uma investigação (manual) posterior, acreditamos que os resultados para os promotores e usuários legítimos são também muito positivos. A pequena fração dos promotores classificados incorretamente foram considerados como *spammers*, ou seja, eles foram estimados, pelo menos, como usuários não-cooperativos. Além disso, a fração de usuários legítimos classificados incorretamente é razoavelmente pequena.

		Predito		
		Promotor	Spammer	Legítimo
Real	Promotor	96.77%	3.23%	0%
	Spammer	3.21%	56.67%	40.13%
	Legítimo	0.09%	8.15%	91.76%

Tabela 2: Classificação com a abordagem híbrida

Analisando os Custos de Detecção de Usuários Não-cooperativos

Até este ponto, foram analisados os métodos de detecção de usuários não-cooperativos (ou seja, nossos novos métodos e o *baseline*) em termos da eficácia da classificação, ou seja, nosso foco foi sobre a efetividade dos resultados da classificação. No entanto, o uso de cada método incorre em custos para o administrador do sistema. Esses custos podem variar dependendo do método utilizado, bem como da coleção de usuários na qual ele é aplicado. Assim, um método que tenha uma boa efetividade não é necessariamente o mais adequado, pois pode incorrer em um custo de utilização muito elevado. Até onde sabemos, não há nenhuma análise prévia, nem estudo de tais custos no problema sendo tratado. Além disso, não temos conhecimento de relatórios públicos disponíveis sobre os custos reais relacionados com a classificação manual de vídeos e de seus usuários, nem sobre os custos associados com os vídeos poluídos que permanecem não detectados e, portanto, são armazenadas por sistemas reais. Assim, nesta seção, nós desenvolvemos um modelo analítico para estimar os custos associados com a utilização de diferentes métodos para identificar os usuários não-cooperantes em SCVOs.

Vislumbramos três diferentes componentes de custo associados à implantação e utilização de um mecanismo para detectar os usuários não-cooperativos: (1) o custo de treino, (2) o custo associado com os usuários não-cooperativos que não são capturados pelo método (falso negativos) e (3) o custo (se houver) associado com os usuários classificados como não-cooperativos. Nós agora introduzimos um modelo analítico desenvolvido para capturar os três referidos componentes de custo. A Tabela 3 resume todas as variáveis usadas neste modelo. Desta forma, o custo de treino pode ser escrito como $c_{Eval} * v_{User} * n_{Train}$, o custo dos usuários não-cooperativos que o método não identificou pode ser escrito como $c_{Pollution} * v_{NC} * n_{FalseLeg}$ e o custo dos usuários preditos como não-cooperativos pode ser escrito como $c_{Eval} * v_{User} * n_{NC}$. Assim, o modelo do custo gerado é:

$$cost = c_{Eval} * v_{User} * (n_{Train} + n_{NC}) + c_{Pollution} * v_{NC} * n_{FalseLeg} \quad (1)$$

Notação	Definição
c_{Eval}	Custo para avaliar manualmente um único vídeo
$c_{Pollution}$	Custo associado com um único vídeo poluído deixado no sistema
n_{Train}	Número de usuários no conjunto de treino
n_{Test}	Número de usuários no conjunto de teste
n_{NC}	Número de usuários classificados como não-cooperativos
$n_{FalseLeg}$	Número de usuários não-cooperativos classificados como legítimos
v_{NC}	Número médio de vídeos postados por cada usuário não-cooperativo
v_{User}	Número médio de vídeos postados por um usuário

Tabela 3: Variáveis envolvidas no modelo de custo

Para realizar nossas análises, criamos diferentes cenários, com base nas variáveis de entrada do modelo analítico, a fim de comparar dois métodos: o nosso melhor método proposto (o método híbrido) e o *baseline* que utiliza todos os dados de treino disponíveis. Em seguida, analisamos os custos totais associados a cada método em cada cenário. Por questões de espaço, apresentaremos aqui apenas um dos resultados mais interessantes que obtivemos. No cenário referente a este resultado, utilizamos duas novas variáveis: C e T. A variável C representa a razão $C = c_{Pollution}/c_{Eval}$ e a variável T indica o tamanho do conjunto de teste como um múltiplo do tamanho do conjunto de teste na nossa coleção de usuários.

Resumimos o impacto das variáveis C e T no custo total associado aos métodos plotando, na Figura 1, a razão R do custo total associado com o método híbrido para o custo total associado ao *baseline* para vários valores de C e T. Note que os valores de R inferiores a 1 implicam em um menor custo total para o nosso método enquanto

os valores de R maiores que 1 implicam em um custo menor para o *baseline*. Como podemos ver na figura, para valores de C iguais ou maiores que 4, ou seja, $c_{Pollution}$ ser pelo menos 4 vezes maior do que c_{Eval}, o método híbrido tem um custo total menor do que o *baseline* para um conjunto de teste até 100 milhões de vezes maior do que é na coleção usada, ou seja, para um conjunto de teste com até 412,5 bilhões de usuários.

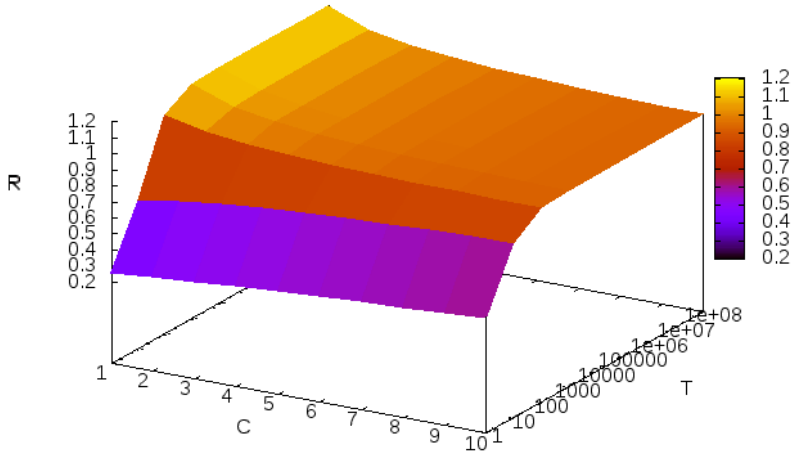


Figura 1: Custos totais em função das variáveis C e T

Conclusões

Nesta dissertação exploramos algoritmos de classificação semi-supervisionados com múltiplas visões para identificar usuários não-cooperativos em SCVOs. Foram avaliadas três abordagens, construídas a partir de dois diferentes métodos para combinação dos resultados das múltiplas visões. Em comparação com métodos supervisionados, nossa melhor abordagem alcançou um custo-benefício muito mais favorável entre identificar usuários não-cooperativos e a quantidade de treino necessária.

Além disso, construímos um modelo de custo para comparar quantitativamente nosso melhor método com o *baseline* utilizado. A análise desse modelo mostrou que, quando o custo de manter um vídeo de poluição no sistema for pelo menos quatro vezes maior que o custo de avaliação manual de um vídeo, nossa abordagem proposta tem menor custo mesmo para bases de teste até 100 milhões de vezes maior que a utilizada neste trabalho. Essa suposição é plausível, já que, na prática, espera-se que realmente o custo de avaliação de um vídeo seja bem menor que o custo de manter um vídeo poluído no sistema.

List of Figures

List of Tables

List of Algorithms

Contents

Chapter 1

Introduction

In this chapter, we discuss the main motivation and arguments that support this work. We also briefly describe our work and explicitly state our contributions.

1.1 The Problem

With the popularization of the Web in the last few years, the number of people that use the Internet is increasingly growing. A significant portion of these users watch online videos. According to comScore [2010a], 84.48% of the Internet audience in the United States watched online videos in March 2010, being responsible for displaying more than 31 billion videos in the period. Because of this demand for online videos, online video sharing systems (OVSSs), such as YouTube and Yahoo! Video[1], are experiencing a vertiginous growth of popularity. Among these sites, YouTube is the one that most stands out, being responsible for providing 41.8% of the total amount of videos watched in the period mentioned above. The fact that users may create their own videos and post them on the Web, passing from the role of viewers of the content to the role of content creators, greatly contribute to the popularity of OVSSs. According to YouTube [2010], every minute, 24 hours of video is uploaded to YouTube. Moreover, according to comScore [2010b], YouTube is the second site in number of queries received in March 2010. Typically, OVSSs have several mechanisms to facilitate the retrieval of videos. One such mechanism is a search engine, where the user types keywords in order to find videos related to an information need. Another mechanism consists of various ordered lists of top-videos, each one sorted according to a certain criterium, such as number of times a video was viewed or number of comments that a video received. Yet another mechanism consists of relationships established among users and/or videos.

[1] http://www.youtube.com, http://www.video.yahoo.com

For example, each user can have a list of friends or a list of favorite videos. Thus, a user can retrieve the favorite videos of her friends very easily. Another example is the ability of a user to respond to a video by posting a related video in response to it, as in the *video-response* feature provided by YouTube. A user is able to watch all the responses posted to some desired video, which supposedly would contain related content also of interest, without having to make a new search[2].

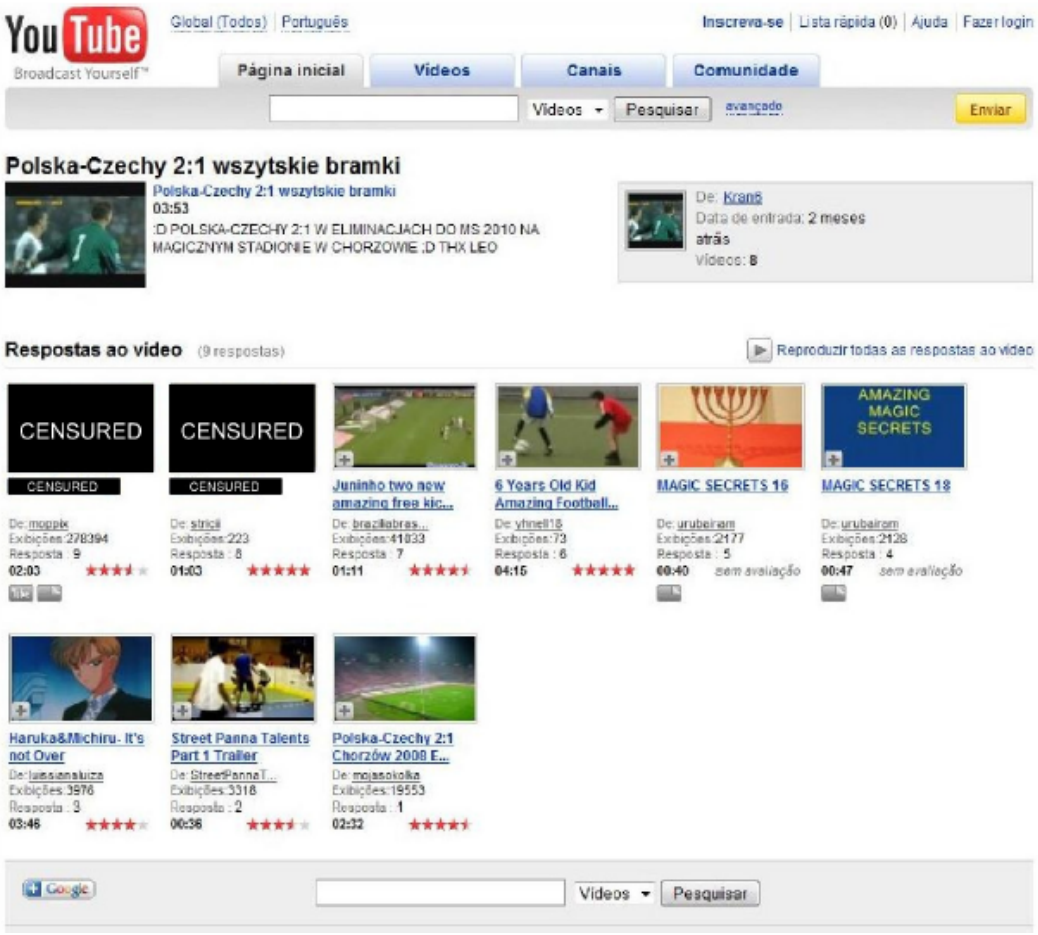


Figure 1.1: Example of spammers

The three aforementioned factors (1) popularization of OVSSs, (2) the possibility for users to post their own videos, and (3) the mechanisms of video retrieval, make room for non-cooperative actions by the users themselves. Previous work has found evidence of non-cooperative users exploiting the video-response feature in YouTube Benevenuto et al. [2009b]. Such users basically post completely unrelated videos in response to previously uploaded videos. One example of such actions is a video con-

[2]YouTube also provides a related video list, associated with each video, which is created according to a proprietary algorithm.

taining advertisement of adult content posted in response to a video of a very popular soccer game. According to Benevenuto et al. [2009a], those non-cooperative users can be classified into two types: spammers and content promoters. Spammers fit exactly in the previous example, since they are users who post unrelated videos in response to popular videos in order to increase the visibility of their own videos, as can be seen in Figure 1.1. A promoter is a user who tries to gain visibility towards her own video by posting a large number of videos, mostly unrelated, in response to it, aiming at boosting the number of video-responses of the target video and making it enter more quickly in the top-list of most responded videos. An example of promoters is shown in Figure 1.2. Similarly, the same types of non-cooperative actions may occur in other features, such as the comments posted by users.

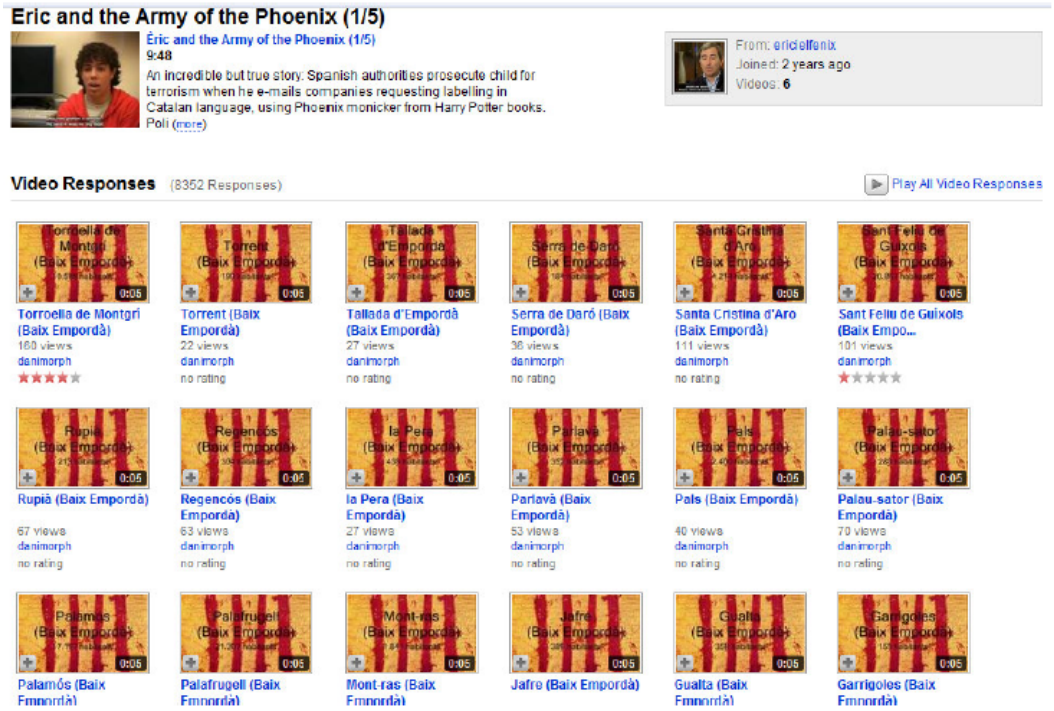


Figure 1.2: Example of promoters

Content pollution brings several disadvantages to OVSSs, including: (1) loss of service effectiveness and credibility, as users, when navigating through the system, may be faced with an unacceptable amount of polluted content, (2) waste of space as the system has to store all the polluted content, (3) waste of bandwidth as users may watch at least a portion of a video to determine that it is pollution, and (4) loss of effectiveness of caches and content distribution networks that the OVSSs employ to replicate popular content (i.e., videos in the top-lists) so as to improve the service

provided to users.

The first and, to the best of our knowledge, unique effort to address the problem of content pollution in OVSSs was done in Benevenuto et al. [2009a]. The authors propose classification-based mechanisms to identify users who are spammers and promoters, differentiating them from legitimate users of the system. Applying a supervised classification algorithm to a collection of 829 pre-classified users, the authors were able to detect the vast majority of the promoters as well as large fraction of the spammers.

Supervised methods need to "learn" a classification function through a set of training data. The drawback of applying such methods to detect non-cooperative users in YouTube is that the manual generation of the training base is very costly, as thousands of videos must be watched. For instance, in Benevenuto et al. [2009a], the authors mention that more than 20,000 videos were manually classified in order to build the collection of 829 users. Moreover, the learning process has to be continuously performed, usually with different training sets, to cope with changes in the strategies adopted by non-cooperative users. Alternatively, unsupervised methods require no training data at all, although the lower cost comes at the expense of a lower classification effectiveness. A better tradeoff between cost and classification effectiveness may be achieved with semi-supervised methods, which combine a smaller amount of labeled data with a large amount of unlabeled data to improve classification.

1.2 Proposed Method

In this work, we explore multi-view semi-supervised classification strategies, which allow us to reduce significantly the amount of training needed to detect spammers and content promoters in OVSSs. The main idea behind multi-view strategies is to use different representations for each element of the problem, so that each representation constitutes a view. For example, in the e-mail domain there are at least two views: (1) attributes extracted from the e-mail content; and (2) attributes extracted from the social network established between senders and receivers. This way, we can use a different classifier for each view and combine the results of each one. Multi-view algorithms rely on the assumptions that the views are compatible and uncorrelated. Intuitively, a problem has compatible views if all examples are labeled identically by each view. On the other hand, two views are uncorrelated if, given the label of any example, its description in each view is independent, i.e., each feature is presented only in one view in a way that the views have disjoint feature sets.

Our proposed methods explore the fact that, in this problem, there is a natu-

ral partition of the feature space in sub-groups (views), each being able to classify a given user when enough training data is available. Thus, the assumptions necessary for multi-view algorithms to work well are satisfied in our context, thus justifying the utilization of this approach. Thus, it is possible to combine the views to allow unlabeled data to be used to augment a much smaller set of labeled samples. We explore two strategies for combining the results from multiple views and selecting which unlabeled samples should be included in the training set. One strategy is based on the agreement of views regarding the label of an unlabeled sample, whereas the other is based on a rank aggregation strategy in which rankings based on the confidence in the classification are combined. We applied our methods to the same user collection used in Benevenuto et al. [2009a]. Our results demonstrate that we are able to reduce the amount of training by a factor of 5 without significant losses in classification effectiveness.

1.3 Costs of Detecting Non-cooperative Users

The use of each non-cooperative user detection method incurs in costs to the system administrator. Such costs may vary depending on the method employed as well as on the user collection on which it is applied. Thus, a method that has a good effectiveness is not necessarily the most suitable, because it may incur in a very high utilization cost. To the best of our knowledge, there is no previous analysis nor study of such costs in the context we are interested. Moreover, we are not aware of publicly available reports on actual costs related to the manual classification of videos and of their users nor on the costs associated with the polluted videos that remain undetected and thus are stored by real systems. Thus, in this work, we develop an analytical model to estimate the costs associated with the utilization of different methods to identify non-cooperative users in OVSSs. Also, we apply this model in different scenarios in order to compare our best proposed method (a hybrid method) with a supervised method which uses all the training data available (our baseline). The results of this analysis showed that our method has a lower cost when compared to the baseline for most of the analyzed scenarios.

1.4 Organization of the Dissertation

This dissertation is organized in seven chapters. The remainder of this work is organized as follows. Next chapter discusses related work. Chapter 3 presents an overview

of our multi-view method, describing the two view combination strategies and the classifier adopted. Chapter 4 describes our evaluation methodology, whereas the most representative results are discussed in Chapter 5. Chapter 6, presents an analytical model developed to estimate the costs associated with the utilization of different methods to identify non-cooperative users in OVSSs. Finally, Chapter 7 offers conclusions and directions for future work.

1.5 Publications

The following publication is a direct contribution of this dissertation. It has been awarded with the José Mauro Castilho award as the best paper of the XXV Simpósio Brasileiro de Banco de Dados, by the Sociedade Brasileira de Computação.

- **Langbehn, H. R.**, Ricci, S, Gonçalves, M. A., Almeida, J. M., Pappa, G. L., Benevenuto, F. A Multi-view Approach for Detecting Non-Cooperative Users in Online Video Sharing Systems. Journal of Information and Data Management, v. 1, p. 313-328, 2010.

Chapter 2

Related Work

In this chapter, we discuss related work. First, we report some efforts related to content pollution in different domains of application. Then, we focus our attention on the specific problem of video pollution. Finally, we describe some related work on multi-view semi-supervised learning.

2.1 Content Polution

Content pollution has been found in various applications and domains. Web spam is a type of pollution that is manifested through the creation of (typically fake) web pages. These spam web pages, typically useless for human visitors, are built so as alter or inflate the results of link analysis algorithms (e.g. PageRank Brin and Page [1998]) used by search engines. The ultimate goal is to mislead search engines into erroneously lead users to certain sites. In Fetterly et al. [2004], the authors propose to use statistical analysis of various web page properties, such as linkage structure and page content, to locate spam web pages. The idea is that certain classes of spam pages, in particular those that are machine-generated, diverge in some of their properties from the properties of "legitimate" web pages. The authors found that outliers in the statistical distribution of these properties are highly likely to be web spam. In Castillo et al. [2007], the authors propose to identify hosts of spam pages through the use of the Web graph topology, by exploiting the link dependencies among Web pages. They characterize the web pages according to attributes taken from the Web graph itself and to text attributes extracted from pages of each host. They found that linked hosts tend to belong to the same class: either both are spam or both are non-spam. Strategies to semi-automatically separate good web pages from spam are proposed in Gyöngyi et al. [2004]. The basic idea is to start with a number of manually classified

good seeds, and then exploit the link structure of the web to discover other pages that are likely to be good as well. The authors obtained good results, showing that their method can effectively filter out spam from a significant fraction of the Web, based on a seed set of less than 200 sites.

In the e-mail domain, a characterization of traffic with the goal of identifying and quantifying properties that distinguish legitimate e-mails from e-mail spams is presented in Gomes et al. [2007]. The authors identify a number of characteristics, such as e-mail arrival process, e-mail size distribution and temporal locality of e-mail recipients, which can be used to separate legitimate traffic from spam, and conjecture that such differences are due to inherent differences in the way legitimate users and spammers behave: whereas the former are typically driven by bilateral relationships, spamming is typically a unilateral action, driven by the goal of reaching as many users as possible. In Xie et al. [2008], the authors use features extracted from the content of e-mails, such as the time when the e-mail was sent, the sender IP address and URLs contained in the e-mail body, to identify URLs that lead to spam web pages as well as IP addresses of botnet hosts. Botnets are programs that are distributed across multiple computers and used to send a large number of spams in a short period of time. The authors developed a spam signature generation framework which does not require pre-classified training data or white lists, and outputs high quality regular expression signatures that can detect botnets spam with a low false positive rate.

In Thomason [2007], the author addresses the presence of spam in blogs, claiming they are due to the combination of three factors, namely, the existence of several means to create a spam on a blog (e.g., blog post, comments, etc), the potential of reaching a large number of people with a single spam, and the limitation of anti-spam technology available at the time for blogs. The author also evaluates the effectiveness of two e-mail anti-spam tools in classifying blog comment spams, and showed that statistical anti-spam solutions developed for e-mails are effective in detecting blog comment spam. Another approach is taken by Lin et al. [2008], who use the temporal dynamics of attributes extracted from the content of blog posts to identify spam blogs. They use three main ideas in their blog spam detection framework. First they represent the blog temporal dynamics using self-similarity matrices. Second, they show that the blog temporal characteristics reveal important attribute correlations, depending on type of the blog. Third, they propose to else use temporal structural properties computed from self-similarity matrices across different attributes. These features are combined with content based features extracted from different parts of the blog, like URLs and post content. To test the method, they use an SVM-based spam blog detector using the proposed features, reaching 90% accuracy on real world data sets.

Some other efforts focus on detecting non-cooperative behavior in online social networks. In Lee et al. [2010], the authors propose and evaluate a honeypot-based approach for uncovering social spammers in online social systems, with the goal of preserving community value. They call honeypots information systems resources that monitor spammers' behaviors and log their information. Their proposed method has two key components: (1) the deployment of social honeypots for harvesting deceptive spam profiles from social networking communities; and (2) statistical analysis of the properties of these spam profiles for creating spam classifiers to actively filter out existing and new spammers. The authors found that the deployed social honeypots identify social spammers with low false positive rates and that the harvested spam data contains signals that are strongly correlated with observable profile features. Based on these features, they used machine learning based classifiers for identifying previously unknown spammers with high precision and a low rate of false positives. In Wang [2010], the author studies Twitter as an example of spam bots detection in online social networking sites. He proposes a machine learning approach to distinguish the spam bots from normal ones. The author uses two kind of features: (1) graph-based features, to explore followers and friend relationship among users; and (2) content-based features, which are extracted from users' most recent 20 tweets. Experiments show that the detection system is efficient and accurate to identify spam bots in Twitter. In Gao et al. [2010], the authors present a study to quantify and characterize spam campaigns launched using accounts on online social networks. They used a dataset of asynchronous "wal" messages between Facebook users and found some interesting results: (1) more than 70% of all non-cooperative wall posts advertise phishing sites; (2) more than 97% of the non-cooperative accounts are compromised [1] accounts, rather than "fake" accounts created solely for the purpose of spamming; and (3) when ajusted to the local time of the sender, spamming dominates the actual wall post activity in the early morning hours, when normal users are asleep.

There are also non-cooperative users in product sites. Lim et al. [2010] propose a behavioral approach to detect review spammers who try to manipulate review ratings on some target products or product groups. They identify several characteristic behaviors of review spammers and model these behaviors so as to detect the spammers. They, also, propose scoring methods to measure the degree of spam for each reviewer and apply them on an Amazon review dataset. Their results show that the proposed ranking and supervised methods are effective in discovering spammers. Also, they show that the detected spammers have more significant impact on ratings compared with

[1] The main account of its owner, which is used also in non-spam tasks.

the unhelpful reviewers.

Many previously proposed strategies for identifying and combating content pollution on the Web are based on evidence extracted from textual descriptions of the content, treating the text as a set of objects with associated attributes, and then using some classification method to identify polluted content Heymann et al. [2007]. A framework to detect spam in tagging systems is proposed in Koutrika et al. [2007]. As tagging systems are gaining popularity, they become more susceptible to tag spam. Misleading tags can be generated in order to increase the visibility of some resources or simply to confuse users. They introduce a framework for modeling tagging systems and user tagging behavior. They also describe a method for ranking documents matching a tag based on taggers' reliability. Using the proposed framework, they study the behavior of existing approaches under non-cooperative attacks and their ranking method. Some other strategies are based on image processing algorithms to detect spam in images. An example of this strategy is presented in Wu et al. [2005] where image attributes are used in conjunction with attributes taken from the text of e-mails to improve e-mail spam detection. The authors analyzed a large collection of spam e-mail containing images and identified a number of useful visual features for this application. Their results showed that the proposed system can add significant filtering power to the existing text based anti-spam filters.

2.2 Video Pollution

The first evidence of the occurrence of non-cooperative behavior on the use of the video-response feature on YouTube was raised in Benevenuto et al. [2009b]. In that article, the authors present a comprehensive characterization of the properties of the YouTube *video-response network*, that is, the network that emerges from video-based user interactions. In Benevenuto et al. [2009a], the same authors further characterize the behavior of three classes of users, namely, legitimate users, spammers and content promoters. They exploit several attributes based on the users' profiles, the users' social behavior in the system (i.e., the relationships established among them) and the videos posted by the users as well as their target (responded) videos to classify users into one of the three classes. Adopting a supervised classification algorithm, they were able to detect the vast majority of promoters (over 95% of accuracy). While a significant amount of spammers were detected, the proposed method also missed a large fraction of them, which were incorrectly considered legitimate users. The false negatives may be due to spammers who exhibit a dual behavior, acting similarly to legitimate users

some of the time.

The results presented in Benevenuto et al. [2009a] leave two venues for further exploration, namely, improving the detection of spammers and reducing the cost of building the training set. The former seems to require the use of content-based techniques to extract semantics and compare pairs of videos. This is outside the scope of this work. Here, we are concerned with the second problem, that is, reducing the cost of building the training data without degrading classification effectiveness.

2.3 Multi-view Semi-supervised Learning

We here cover some work on multi-view semi-supervised learning. Our goal is not to exhaustively cover the literature but only to describe the most related efforts.

As previously mentioned, supervised classification algorithms require a training phase in which all examples must be previously manually labeled. This classification is very costly and requires the involvement of a large number of people to be held on time. In Blum and Mitchell [1998], the authors present a multi-view approach for classification of web pages, where the labeling cost is reduced. However, the proposed approach assumes that there is total agreement in the classification performed from each view, which may not always be the case. A less constrained multi-view classification approach is proposed in Christoudias et al. [2008]. The authors use a conditional entropy criterion to detect differences in the predictions of the classifiers. When divergence is identified in any sample, the sample is filtered, i.e., removed from the unlabeled data set and placed in a data set that will not be used. After this process, none of the samples in the unlabeled data set has divergences and, therefore, the multi-view classification approach can be used normally. The work described in Hovelynck and Chidlovskii [2010] is the only one we were able to find that exploits multi-view classification to solve problems involving social networks aspects. Particularly, the proposed method is applied to the problem of distinguishing responsive documents (i.e., those relevant in the context of a legal case) in a corpus of e-mails (Enron Corpus). The authors consider two views: (1) the text representation of the e-mails; and (2) the social network that is implicit in the group of people communicating with each other. In a very recent work (Perez et al. [2011]), the authors exploit the Borda Count method to combine results from different classifiers in the context of face recognition. Borda Count is a method to agregate ranked lists (Black [1963]) that was previously proposed to combine lists of candidates in elections. More details about this method is given in Section 3.2. The authors make some improvements in the Borda Count algorithm through the use of

a weighted count and a threshold to eliminate low scores from the process. As far as we know, this is the only other approach to exploit the Borda Count method for multi-view learning.

We here extend the work presented in Benevenuto et al. [2009a] by proposing multi-view classification approaches to reduce the labeling cost for detecting non-cooperative users in OVSNs. As in Christoudias et al. [2008], our approaches consider that there may be divergence in the classifications of each view. We here deal with divergence adopting two strategies, namely: (1) considering only elements for which all views agreed in the classification, i.e., disregarding elements whose classification diverged and (2) exploring the confidence of each view's prediction and taking, among elements whose classifications diverged, only those with the largest aggregated confidence for a given class using certain ranking aggregation strategies. Moreover, we also propose hydrid solutions combining both strategies. These solutions are much simpler than the one adopted in Christoudias et al. [2008] and, as will be shown in Chapter 5, lead to very good results.

Chapter 3

Multi-View Semi-Supervised Approach to Detect Non-Cooperative Users

In this chapter, we present an overview of our multi-view semi-supervised approach to detect spammers and content promoters on OVSSs (Section 3.1). Then, we present our proposed methods to combine the results from different views, a step required by the semi-supervised algorithm (Section 3.2). Finally, we briefly present the classifier we used in each view (Section 3.3).

3.1 Overview of our Approach

The main idea behind our proposed multi-view semi-supervised classification is start with a small set of labeled data as the training set while still being able to expand it with examples extracted from a set of unlabeled data. The expanded training set is then used to classify the desired objects. During the insertion of new elements into the training set, the training set and the unlabeled set are partitioned into views. Each view consists of a number of attributes of the elements in the training set. In other words, each element in the training set is present in all views, represented by a different set of attributes in each of them. Take, for instance, the classification of YouTube users performed in Benevenuto et al. [2009a]. We can say that the authors explore three different views of each user, namely: the user profile, her social network, and the objects owned by or target by her. When no more elements can be inserted into the training set, all the views of the final training set are integrated into a single

one, which is used as the final training data used in the classification. The idea is that, to work, the predictions of the classifiers trained with data from each view (hereafter also called the "views") should be compatible, i.e., all samples are labeled identically by all or most of the views, and the elements' representations in the views should be uncorrelated, i.e., each element is described by a different and disjoint set of attributes in each view.

This introduction of examples into the training set occurs through an iterative process, according to Algorithm 1. In each iteration of the algorithm, the classifier of each view v is trained with its corresponding training set L_v, and is used to predict the class of each element u_v^m in the unlabeled dataset U_v. Along with the predictions, the evaluation step also gives the confidence $\theta_v^m(k)$ of each element u_v^m being of class k. The confidence will be better explained in Section 3.3, where the classifier is presented. Based on the predictions of each classifier, a method is used to determine whether there are unlabeled elements that can be inserted into the training set and to give the final predicted class for each of these elements, which is done by the GetNewInsertions function. As an input, the GetNewInsertions function also uses the fraction of elements belonging to each class in the original training set, computed in lines 3-5 of the algorithm. This can guide some policies about how many elements from each class should be incorporated as training data, as we shall see in Section 3.2. Then, selected items have their class updated to the final predicted class and are incorporated into the training set and removed from the unlabeled data set. In other words, each selected example u_v^m has its class updated and is removed from the unlabeled data set of each view U_v and inserted into the training set of each view L_v. This process is repeated until either no more elements can be inserted into the training set or the unlabeled data set is empty. The methods used for the selection of elements to be included in the training set define the view combination strategy adopted. We discuss the strategies considered in this work in the next section.

For the specific problem of detecting non-cooperative users on OVSSs, our proposed multi-view semi-supervised approach can be applied to: (1) decrease the amount of training data needed by the classifier as well as (2) increase the quality of the classification process. Semi-supervised approaches can start with much smaller training set than the ones used in supervised approaches, since they can add more examples to the training set by applying the iterative process described in Algorithm 1. Indeed, semi-supervised approaches require only enough examples from each class to allow the classifiers to execute (reasonably) well in the first iteration of the algorithm, and select new examples to be introduced in the training set. How many examples is enough for the classifier depends on a variety of factors, like how discriminative the attributes of

Algorithm 1 Multi-view Semi-supervised Training Expansion

Input: The number of views V, a set C of classifiers C_v for each view v ($v = 1...V$), the number of classes K, the number of elements M in the unlabeled data set, a set L of labeled samples l_v^n with the attributes from each view v ($v = 1...V$) for each element l^n ($n = 1...N$), and a set U of unlabeled samples u_v^m with the attributes from each view v ($v = 1...V$) for each element u^m ($m = 1...M$). {N is the number of elements in the labeled data set}

Output: Set L expanded to include all initial elements as well as new elements added by the algorithm.

1: **repeat**
2: $insertion \leftarrow FALSE$
3: **for** $k \leftarrow 1$ to K **do**
4: $B_k \leftarrow$ the fraction of elements from class k in L
5: **end for**
6: **for** $v \leftarrow 1$ to V **do**
7: Train C_v on L_v
8: Evaluate C_v on U_v giving predictions in P_v and confidences in θ_v {θ is a set of the confidences $\theta_v^m(k)$ from each view v of the pertinence of element u^m to class k}
9: **end for**
10: {I receives elements that can be added to the training data along with the predicted class for each element}
11: $I \leftarrow \text{GetNewInsertions}(V, K, B, M, UP, \theta)$
12: **if** $|I| > 0$ **then**
13: $insertion \leftarrow TRUE$
14: **end if**
15: **for** $i \leftarrow 1$ to $|I|$ **do**
16: **for** $v \leftarrow 1$ to V **do**
17: $L_v \leftarrow L_v \cup u_v^i \in I$
18: $U_v \leftarrow U_v \setminus u_v^i \in I$
19: **end for**
20: **end for**
21: $M \leftarrow |U|$ {Update the number of elements in the unlabeled data set}
22: **until** ($|U| = 0$ or $insertion = FALSE$)

the examples given are.

Our approach can also help improving the classification effectiveness by letting us introduce a 4^{th} view - a video content view - composed of attributes extracted from the video itself. As discussed in Section 2, this might improve the detection of spammers. However, given that we do not have access to a test collection with video content based attributes, we leave this task for future work. In this paper, our focus is on applying the proposed approach to reduce as much as possible the amount of training data required by the classifier without loss in the quality of the classification.

3.2 Combining Results from Multiple Views

An important step of the multi-view semi-supervised approach is how to select the elements to be inserted into the training set on each iteration, i.e., function *GetNewInsertions* in Algorithm 1. This function is responsible for combining the classification results of different views, that is, assigning a label to each element, and then selecting those, according to certain criteria, that should be added to the training set. Elements that are not selected remain as part of the unlabeled data set. Several strategies can be adopted to perform this task. We here explore two such strategies: one is based on view agreement and the other is based on a method to aggregate ranked lists called Borda Count (Black [1963]). The view agreement approach was chosen because of its simplicity and possibility of obtaining good results when the concordance between the views is high enough. However, this approach may not work well when the concordance between views is low. Because of this, we also chose to use an approach based on Borda Count algorithm. Borda Count uses each view separately, generating a rank for each view. Only in the end of the process the ranks are aggregated. Thus, borda count is expected to work well even when the concordance between views is low. We do not further investigate other combination strategies since our focus is to find a method that works well in our context, not to find the best method for the problem in hand. As we will see in Chapter 5, these two approaches satisfy this criterium.

Before introducing each strategy, we first address one of the criteria used to select elements to be inserted into the training set, which is applied by both strategies. During our initial experiments we found that it is very important, for the sake of classification effectiveness, to keep the distribution of the number of labeled elements per class roughly stable as new elements are inserted into the training set. In the specific problem of classifying non-cooperative users in OVSSs, user collections (and the one used here in particular) tend to be very skewed Benevenuto et al. [2009a] as

most users tend to be legitimate. Thus, there is a natural bias towards the larger class. If we do not keep the class distribution roughly stable in the training set, this bias will tend to increase even more, compromising the classification effectiveness for the smaller classes, which is, in the specific case, what we care most. Therefore, we compute the initial distribution of elements across classes in the training set, specified as the fractions B of elements in each class, and use it to constrain the insertion of new elements in each iteration.

Algorithm 2 GetNewInsertions_ViewAgreement

Input: The number of views V, the number of classes K, a set B with the initial fraction B_k of elements from each class k, $k = 1...K$, the number of elements M in the unlabeled data set, a set U of unlabeled samples u_v^m with the attributes from each view v, $v = 1...V$, for each element u^m, $m = 1...M$, a set P of the class predictions P_v^m from each view v, $v = 1...V$, for each element u^m, $m = 1...M$, and a set θ of the confidences $\theta_v^m(k)$ from each view v of the pertinence of element u^m to class k.

Output: Set I with the elements selected to be inserted along with the predicted class of each element.

1: **for** $k \leftarrow 1$ to K **do**
2: $S_k \leftarrow \{\}$ { S_k will contain candidate elements of class k to be considered for insertion}
3: **end for**
4: **for** $m \leftarrow 1$ to M **do**
5: **if** $P_r^m = P_s^m, \forall r, s \in V, r \neq s$ **then**
6: {All views agree on class predicted for element u^m}
7: $k \leftarrow P_r^m$ {k gets the class predicted by all views for element u^m}
8: $S_k \leftarrow S_k \cup u^m$
9: **end if**
10: **end for**
11: $I \leftarrow$ GetElementsByConfidence(B, θ, S) {Select elements from each S_k with largest general confidence, constrained by class distribution B}

We now describe each considered strategy to combine the results from all views. The View Agreement strategy works as shown in Algorithm 2. For each element u^m in the unlabeled data set, u^m is selected provided that all views agree on the class predicted for it (say, class k). If selected, u^m is inserted into the set of candidate elements for the predicted class, S_k. Note that we can not insert u^m directly into the I set, or else we might change the distribution of elements across classes in the training set. After the candidate selection process finishes, we use the confidence of the predictions θ and the target class distribution B to determine which elements from the sets of candidate S will be inserted into I, what is done by the GetElementsByConfidence function. We do so by choosing the elements from each S_k with largest general confidence, constraining

the number of selected elements from each set so as to keep the distribution of elements across classes as close as possible to B. If any class does not have any agreement, or if the number of agreements is too small to keep the classes distribution unchanged, what is measured by the size of each S_k, the GetElementsByConfidence function returns an empty set to I, which usually implies in stop the whole process. We can calculate the general confidence in the prediction of each element u^m in several ways. We here use the sum of the confidences of each view for the predicted class. For example, if there are $V = 2$ views, and the class predicted (by both views) for element u^m is k, then the general confidence θ^m in the prediction of element u^m will be the confidence of the view 1 for element u^m being from class k plus the confidence of view 2 for element u^m being from class k, that is $\theta^m = \theta_1^m(k) + \theta_2^m(k)$. Alternatively, we could weight the confidence from each view by a factor reflecting the trust we have in it.

Algorithm 3 GetNewInsertions_BordaCount

Input: The number of views V, the number of classes K, a set B with the initial fraction B_k of elements from each class k, $k = 1...K$, the number of elements M in the unlabeled data set, a set U of unlabeled samples u_v^m with the attributes from each view v, $v = 1...V$, for each element u^m, $m = 1...M$, a set P of the class predictions P_v^m from each view v, $v = 1...V$, for each element u^m, $m = 1...M$, and a set θ of the confidences $\theta_v^m(k)$ from each view v of the pertinence of element u^m to class k.

Output: Set I with the elements selected to be inserted along with the predicted class of each element in I.

1: $I \leftarrow \{\}$
2: **for** $k \leftarrow 1$ to K **do**
3: $Q_k \leftarrow 0$ {Q_k will contain the total aggregated ranking values Q_k^m for each element u^m, produced by all views for class k. Q_k values are initially set to zero.}
4: **for** $v \leftarrow 1$ to V **do**
5: $sortedU_v(k) \leftarrow$ Sort U by $\theta_v(k)$, in ascending order
6: **for** $m \leftarrow 1$ to M **do**
7: $Q_k^m \leftarrow Q_k^m$+GetRankingValue(u_k^m,$sortedU_v(k)$) {Aggregate ranking values for u_k^m obtained with all views for class k}
8: **end for**
9: **end for**
10: Sort U by Q_k values, in descending order {Sort U according to final aggregated ranking values}
11: $S_k \leftarrow$GetElementsByRank(B_k, U, P) {Get the first elements of U sorted by the aggregated ranking values in Q_k keeping the initial proportion of classes}
12: $I \leftarrow I \cup S_k$
13: $U \leftarrow U \setminus S_k$
14: $M \leftarrow |U|$ {Update the number of elements in the unlabeled data set}
15: **end for**

The Borda Count strategy is based on a method previously proposed to combine list of candidates in elections and later used to solve computational problems such as the aggregation of rankings produced by multiple search engines Dwork et al. [2001]. It was thus originally proposed in a very different context, and, to the best of our knowledge, this is the first work that explores its use in the context of multi-view classification. The advantage of this strategy is that is more resilient to issues related to the magnitude of the absolute values of the confidence in the prediction of elements in classes, which for some classes may be very low/high or may not help distinguish much between several classes (e.g., close confidence for all classes). These issues may affect any method which rely on the absolute values of the confidence (e.g., summation, average, max, min, etc). The strategy is presented in Algorithm 3. It calculates K ranking aggregation values for each element u^m in the unlabeled data set. Each such value corresponds to the aggregation of the rankings generated by each view v with respect to each class k ($k = 1...K$). The algorithm considers one class at a time and, for each element, it gets the confidence of the prediction of each view with respect to the class under consideration. Then, for each view, the algorithm sorts the elements, in increasing order, according to the confidence of the view for that class (line 5), so that the last element is the one with the highest confidence in that view for that class. The sorted elements get a ranking value according to the position occupied by them in the rank (function GetRankingValue), with elements in last positions getting higher values. These values are summed up for all views (line 7) to give a final value of the aggregation of the different ranks for a class. This final value is used to sort the elements again, in descending order (line 10). The top elements of this final rank, i.e., the ones with the higher rank aggregation values, which had its class predicted as k for at least one view, are selected to be inserted into the training data set in a way that the distribution of elements across classes B is not changed, what is done by the GetElementsByRank function (line 11). Like with the View Agreement strategy, a weighted sum of the ranking values could also be applied.

Notice that, for the sake of simplicity, both Algorithms 2 and 3 present the computation of set I for *all classes*. However, the set of candidate elements of each class k selected to be added to the training data, S_k, is determined independently for each class. Thus, different strategies can be applied to different subsets of the classes, depending on the characteristics of the problem being solved. In fact, we do explore such a hybrid approach in our experiments described in Chapter 5, as we shall see. In the following section, we present a brief description of the classifier used in our proposed multi-view method.

3.3 The Classifier

We use Lazy Associative Classification (LAC) (Veloso et al. [2006]) as our classifier. Usually, the training data has strong associations between attribute values and classes, which can be used for the sake of predicting classes for elements. These associations may be hidden in the data meaning that they need to be explicitly revealed. LAC exploits this fact by trying to uncover such associations, with the goal of improving the classification effectiveness.

LAC produces a classification function composed of rules $X \rightarrow k$, indicating the association between a set of attribute values X and a class k. In the following, we denote as R an arbitraty rule set. Similarly, we denote as R_k a subset of R that is composed of rules predicting class k. A rule $X \rightarrow k$ is said to match element u^m if $X \subseteq u^m$, (i.e., element u^m contains all attribute-values in X) and this rule is included in R_k^m. That is, R_k^m is composed of rules predicting class k and matching element u^m. As we can note, $R_k^m \subseteq R_k \subseteq R$.

LAC learns the classification function in two broad steps:

- **Demand-Driven Rule Extraction**: During the rules extraction, the number of rules obtained from the training data may be too large. To avoid this problem, LAC extracts rules on a demand driven fashion, i.e., it uses information about elements in the test set to project the search space for rules. In other words, LAC filters the training data according to the attribute-values of element u^m in the test set, and extracts rules from this filtered training data. Thus, only rules that carry information about element u^m are extracted from the training data, drastically bounding the number of possible rules, what allows an efficient rule extraction.

- **Prediction**: Some rules have stronger associations than others. To measure the strength of an association, LAC uses a statistic called confidence (Agrawal et al. [1993]). The confidence of the rule $X \rightarrow k$ is given by the conditional probability of k being the class of element u^m, given that $X \subseteq u^m$.

  Using a single rule to predict the correct class may be prone to error. Instead, the probability of k being the class of element u^m is estimated by combining rules in R_k^m. More specifically, each rule $X \rightarrow k \in R_k^m$ is a vote given by features in X for class k. The weight of a vote $X \rightarrow k$ depends on the strength of the association between X and k, which is given by $\theta(X \rightarrow k)$. To estimate the probability of k being the class of element u^m, the weighted votes for k are summed and the result is averaged by the total number of votes for k, as expressed by the score function

shown in Equation 3.1 (where $r_x \subseteq R_k^m$ and $|R_k^m|$ is the number of rules in R_k^m). Thus, $s(k, u^m)$ gives the average confidence of the rules in R_k^m. Obviously, the higher the confidence, the stronger the evidence of class membership.

$$s(k, u^m) = \frac{\sum_{x=1}^{|R_k^m|} \theta(r_x)}{|R_k^m|} \tag{3.1}$$

The estimated probability of k being the class of element u^m, denoted as $\hat{\mathrm{p}}(k|u^m)$, is simply obtained by normalizing $s(k, u^m)$, as shown in Equation 3.2. A higher value of $\hat{\mathrm{p}}(k|u^m)$ indicates a higher likelihood of k being the class of element u^m. The class associated with the highest likelihood is finally predicted as the class for element u^m.

$$\hat{\mathrm{p}}(k|u^m) = \frac{s(k, u^m)}{\sum_{l=1}^{K} s(l, u^m)} \tag{3.2}$$

We note that in Algorithms 1, 2 and 3, we denoted by $\theta_v^m(k)$ the confidence of view v on predicting class k for element u^m. Thus, $\theta_v^m(k)$ is indeed the value of $\hat{\mathrm{p}}(k|u^m)$ for view v's classifier.

Chapter 4

Evaluation Methodology

In this chapter, we describe how the multi-view semi-supervised classification method, presented in the previous chapter, is applied to detect non-cooperative users, namely spammers and content promoters, in OVSSs. We start by describing, in Section 4.1, our user test collection, introducing the views considered by our approaches and their associated attributes. Next, the metrics used in the evaluation of our solutions are introduced in Section 4.2, whereas the experimental setup is presented in Section 4.3.

4.1 User Test Collection

In order to evaluate our proposed approaches to identify spammers and content promoters in OVSSs, we need a test collection composed of users of the target system, which in our case is YouTube. In this collection, all users must be pre-classified into legitimate users, spammers or promoters. The process of building such a collection is very expensive, as it requires human effort in watching a potentially very large number of videos. Thus, we here use the same user collection presented in Benevenuto et al. [2009a], which was built primarily through a crawling of YouTube followed by a selection of a subset of the crawled users to be manually classified.

The crawling phase consisted of collecting a sample of users involved in interactions through the use of YouTube video-response, i.e., users who had posted or received video-responses. This crawling, performed in January 2008, gathered a total of 264,460 users, 381,616 responded videos and 701,950 video-responses. Users were gathered by starting with a number of seeds and following their interactions via video-responses (i.e., snowball strategy), thus building, at the end, a video response user network.

In the manual classification phase, a selected user was labelled as a “spammer" if she posted at least one video-response that was considered unrelated to the responded

video. She was labeled as “promoter" if she posted several videos in response to a responded video with the aim of promoting this responded video. A user who is neither promoter nor spammer was labeled as “legitimate". The user test collection built from the manual classification has a total of 829 users, consisting of 641 legitimate users, 157 spammers and 31 promoters. These users have posted 20,644 video-responses to 9,796 responded videos. This collection has the following characteristics: (1) it has a significant number of users from all three classes, (2) it includes, but is not limited to, non-cooperative users with aggressive strategies, as these are the users who generate the most pollution in the system and (3) includes legitimate users with different behaviors. In the following ,we present the user attributes gathered in our collection as well as the multiple views extracted from these attributes and adopted by our classification approaches.

4.1.1 Collection Attributes

Legitimate users, spammers and promoters have different goals in the system and are, thus, expected to act differently while using the system. Such differences may be captured by exploring a number of attributes that reflect how each user uses the system. In particular, our user test collection contains a total of 60 attributes per user, which can be divided into three groups: video attributes, user attributes and attributes of the social network established among the users through the use of the video-response feature.

The video attributes associated with a user relate to features of the videos posted by her as well as the videos responded by her (i.e., the videos that were target of her video-responses). The video features considered are: the duration, the number of views, the number of comments received, ratings, number of times that the video was selected as favorite, number of honors, and number of external links from the video. Note that these attributes serve as indicators of the quality of a video, as perceived by the user community. Three groups of these attributes were created. The first group contains aggregated information from all the videos posted by the user, which may indicate how others see the contributions of this user. The second group contains information only of the video responses posted by the user, which are precisely those videos that can be pollution. The latter group considers only responded videos to which the user posted video-responses. For each of these groups were considered the sum and average of each attribute, totaling 42 video attributes.

The user attributes consist of individual features of user behavior, extracted from the user’s profile on the system. Ten attributes are used: number of friends, number

of videos uploaded, number of videos watched, number of videos added as favorites, number of video responses posted, number of video-responses received, number of subscriptions, number of subscribers, average time between uploads and maximum number of videos uploaded within 24 hours.

The attributes of the user's social network capture the social relations established among users through video-responses, which is one of the several social networks that emerge among users on YouTube. This network is modelled as a directed graph, where each node represents a user, and a edge (i, j) indicates that the corresponding user u_i posted at least one video in response to some video of user u_j. The 5 attributes of social network included in our user collection are: clustering coefficient, betweenness, reciprocity, assortativity and UserRank.

The clustering coefficient of node i, $cc(i)$, is the ratio of the number of existing edges between i's neighbors to the maximum possible number, and captures the communication density between the user's neighbors. The betweenness is a measure of the node's centrality in the graph, i.e., nodes appearing in a larger number of shortest paths between any two nodes have higher betweenness than othersNewman and Park [2003]. The reciprocity $R(i)$ of node i measures the probability of the corresponding user u_i receiving a video-response from each other user to whom she posted a video-response, i.e., $R(i) = \frac{|OS(i) \cap IS(i)|}{|OS(i)|}$, where $OS(i)$ is the set of users to whom u_i posted a video-response, and $IS(i)$ is the set of users who posted video-responses to u_i. Node assortativity is defined, as in Castillo et al. [2007], as the ratio between the node (in/out) degree and the average (in/out) degree of its neighbors. Node assortativity was computed to the four types of (in/out)degree-(in/out)degree correlations. The PageRank Brin and Page [1998] algorithm, commonly used to assess the popularity of a Web page, was applied to the video-response user graph built from the collection. The computed metric, called UserRank, indicates the degree of participation of a user in the system through interactions via video-responses. In total, 8 social network attributes were used.

4.1.2 Multi-View Collection

Unlike in Benevenuto et al. [2009a], where authors applied a single-view supervised classification method, our goal here is to explore multi-view semi-supervised approaches. Thus, we need to extract different views from the user collection. As explained in the previous section, the collection already has three separate groups of attributes, namely user attributes, video attributes and social network attributes. Thus, we take this inherent categorization of the user attributes in the collection to

generate a video view, a user view and a social network view. Each view includes only the attributes of the corresponding group. In other words, the video view consists of 42 attributes, the user view has 10 attributes, and the social network view has 8 attributes.

4.2 Evaluation Metrics

We evaluate the classification approaches by comparing mainly the confusion matrices Kohavi and Provost [1998] produced by each of them. A confusion matrix is illustrated in Table 4.1. Each element in position (i, j) of this matrix represents the percentage of users from class i (i.e., row i) that were predicted, by the classification, as being of class j (i.e., column j). We choose to focus our evaluation on these matrices because they better expose the tradeoffs between correctly classifying users of one class at the expense of misclassifying users of the others. These tradeoffs are particularly interesting for the specific task of classifying YouTube users as either legitimate or non-cooperative. We envision our approaches being used to help system administrators by "flagging" suspicious users for further (possibly manual) investigation. In that case, we believe that it is preferable to improve the detection of non-cooperative users even if it comes at the expense of misclassifying some legitimate users as non-cooperative. These wrongly classified legitimate users will have the chance to be cleared out later. In contrast, non-cooperative users who are misclassified as legitimate may escape undetected as manual investigation of the large number of (predicted) legitimate users is highly unlikely. Thus, when comparing the classification approaches in Section 5, we focus mainly on the confusion matrices, thus allowing us to better assess the tradeoffs between correctly classifying legitimate and non-cooperative users.

In addition to the confusion matrices, we also consider the F1 metric Yang [1999], commonly used to evaluate information retrieval tasks. F1 is defined as a function of precision and recall. The precision (p) of a class k is the ratio of the number of users correctly classified to the total number of users predicted to be of class k (e.g., $p_{spammer} = e/(b + e + h)$ in Table 4.1). The recall (r) of a class k is the ratio of the number of users correctly classified to the number of users in class k (e.g. $r_{spammer} = e/(d + e + f)$). The F1 metric is the harmonic mean between both precision and recall, defined as $F1 = 2pr/(p + r)$. There are two variations of F1, namely Micro-F1 and Macro-F1. Macro-F1 values are computed by first calculating F1 values for each class in isolation, as illustrated above for spammers, and then averaging over all classes. Therefore, Macro-F1 considers equally important the classification effectiveness in each

class, independently of the relative size of the classes, being thus more adequate when the class distribution is very skewed. Since our user test collection is inherently very skewed towards legitimate users[1], we consider only Macro-F1 in our evaluation.

		Predicted		
		Promoter	Spammer	Legitimate
True	Promoter	a	b	c
	Spammer	d	e	f
	Legitimate	g	h	i

Table 4.1: Example of confusion matrix

4.3 Experimental Setup

We ran a series of experiments with five classification approaches. Three approaches are based on the proposed multi-view semi-supervised method, and explore the two view combination strategies introduced in Section 3.2. To evaluate the effectiveness of these approaches, we also consider two baselines. The first one is a single-view supervised method using all the training data available. The comparison against this baseline allows us to evaluate the tradeoff between amount of labeled data and classification effectiveness. As a second baseline, we consider the same single-view supervised method which takes the same amount of labeled data as the proposed strategies. The comparison against this second baseline allows us to evaluate the impact on the classification of incorporating new examples into the training set. Note that we do not compare our proposed multi-view semi-supervised approaches with other semi-supervised methods, leaving this for future work. Note also that here our goal is to find a method capable of reducing the amount of training needed without significant losses in the quality of the classification, what we achieve with the utilization of this three proposed multi-view semi-supervised approaches.

All five classification approaches use the LAC classifier (Section 3.3) and our user collection (Section 4.1) consisting of 60 attributes. For the multi-view semi-supervised approaches, we separated these attributes into 3 views, as discussed in Section 4.1.2. Note that, unlike in Benevenuto et al. [2009a], where the authors used a Support Vector Machine (SVM) classifier, we here choose to use LAC. This choice is mainly because the estimated probabilities of a user being in a class, essential to the multi-view approaches, were found to be much more reliable, according to some initial experiments comparing

[1] We do expect any other user collection of the same type, being representative of the entire user population, to contain a much larger number of legitimate users than of spammers and promoters.

SVM and LAC. Moreover, we also found that LAC achieved somewhat better results, when 100% of training data is used.

The classification experiments were performed using a 5-fold cross validation. The original sample is partitioned into 5 sub-samples. In each test, four of the sub-samples are used as training data and the remaining one is used as test data. For the semi-supervised approaches, the training data is further partitioned into labeled and unlabeled data sets, and used as explained in Section 3.1. This process is repeated 5 times, with each of the 5 sub-samples used exactly once as the test data. The entire 5-fold cross validation is repeated 5 times, using different seeds to shuffle the original data set. Thus, the classification results reported in the next section for each considered approach are averages of 25 runs.

Chapter 5

Experimental Results

This chapter presents the most relevant results of our comparison of the different classification approaches considered. All reported results are averages of 25 runs, as explained in the previous chapter. In all experiments, the test sets (one fold per run) are kept the same for all evaluated approaches. Reported results have standard deviations under 5% of the means.

5.1 Initial Analyses

As explained in Section 4.3, for experimental purposes, when evaluating the multi-view semi-supervised approaches, we need to partition the original training data in each experiment (i.e., 4 folds) into labeled and unlabeled data in order to simulate the situation in which we have a small amount of labeled data for a large amount of unlabeled samples. The goal is to achieve the best tradeoff between the amount of training data, which should be minimum, and the effectiveness of the approach, which should be as close as possible to that of the supervised method using all the available training data. In order to evaluate this tradeoff, we ran a set of initial experiments with our proposed multi-view strategies, increasingly reducing the percentage of the original training data provided as labeled samples for the strategies, while leaving the remaining amount as unlabeled. We tested various percentages (from 40% to 10%), and found that using only 20% of the original training data (and leaving 80% of it as unlabeled) led to the best tradeoff, as shown in Tables 5.1, 5.2, 5.3 and 5.4.

		Predicted		
		Promoter	Spammer	Legitimate
True	Promoter	94.84%	3.23%	1.94%
	Spammer	4.10%	55.26%	40.64%
	Legitimate	0%	5.58%	94.42%

Table 5.1: Classification with 40% of Training Data

		Predicted		
		Promoter	Spammer	Legitimate
True	Promoter	95.48%	3.23%	1.29%
	Spammer	3.33%	54.74%	41.92%
	Legitimate	0.06%	6.18%	93.76%

Table 5.2: Classification with 30% of Training Data

		Predicted		
		Promoter	Spammer	Legitimate
True	Promoter	96.77%	3.23%	0%
	Spammer	3.21%	56.67%	40.13%
	Legitimate	0.09%	8.15%	91.76%

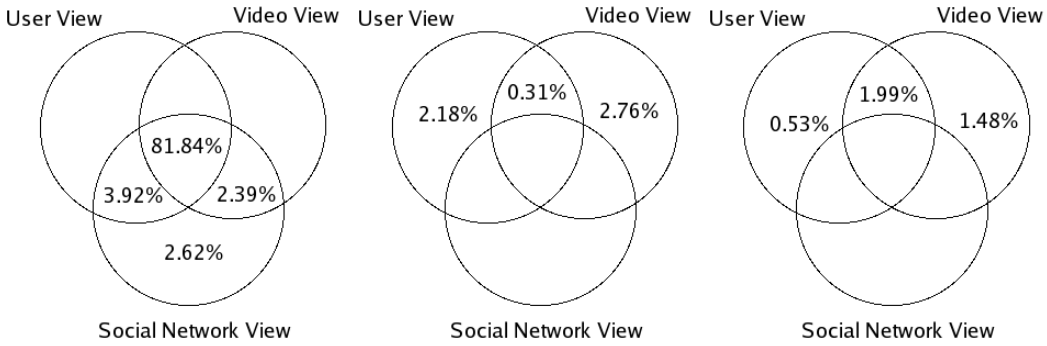
Table 5.3: Classification with 20% of Training Data

		Predicted		
		Promoter	Spammer	Legitimate
True	Promoter	52.26%	43.23%	4.52%
	Spammer	0.64%	55.90%	43.46%
	Legitimate	0.03%	6.77%	93.20%

Table 5.4: Classification with 15% of Training Data

We can see that 20% of the original training set is the smallest amount of labeled data required to produce competitive results in comparison with the supervised approaches. When we further reduce it to 15% of the original training set, the fraction of correctly classified promoters drops sharply to only 52%. The classifier requires a minimal amount of examples of each class to learn the needed patterns to identify specific types of behavior. If a particular class does not have enough examples, the classifier will not have enough information about the elements of this class, thus incurring in misclassifications. When we used only 15% of the original training set as labeled data, the number of promoters, the smallest class, used for learning, was very small (just three examples), compared with the amount of promoters when we used 20% of the training set (five examples). Thus, in the first iteration of the semi-supervised algorithm, there

were incorrect insertions of promoters into the training set, i.e., promoters are inserted as erroneous examples of other classes. This error caused the number of incorrect insertions of promoters to increase further over the additional iterations, which explains the verified loss in the effectiveness of the classifier, mostly due to the misclassified promoters. Next, we focus our comparison on the results when 20% of the original training set is used as labeled data by the multi-view semi-supervised approaches and by the second baseline.



(a) Legitimate Users (b) Spammers (c) Promoters

Figure 5.1: Prediction of classes by each view

Before presenting our classification results, we note that, for both multi-view approaches explored in this work, only two views were considered when introducing new examples into the training data: the user view and the video view. The social network view was disregarded because initial classification experiments applying each view in isolation indicated that it is insufficient to distinguish between different user classes, presenting very poor classification effectiveness. These results are illustrated in Figure 5.1, which shows the intersections, percentage-wise, of the predictions of each view in the first iteration of the multi-view semi-supervised method on the unlabeled set. Figure 5.1a), which refers to the performance on predicting legitimate users, shows that all predictions are inside the social network view, regardless of the predictions of the other views. This means that the social network view predicts all users as legitimate users. The large percentage found in the intersections of the three views in Figure 5.1a) reflects the fact that most users are in fact legitimate, which makes it easier to predict for this class. More importantly, in Figures 5.1b) and 5.1c), all predictions are outside the scope of the social network view, implying that it is not able to predict any user as either promoter or spammer. Obviously, there is no agreement with the other views regarding these classes. We note that, while the social network view was disregarded

while adding new elements into the training data, all 60 attributes are used in the final classification of the test sets. The same holds for the two (supervised) baselines.

5.2 Baseline Results

We start our analyses by considering the performance of our baselines, i.e., classifiers trained with all training data available (Table 5.5) and with the same 20% used by the multi-view approaches (Table 5.6). For the first baseline, promoters and legitimate users are classified correctly in almost 100% of the cases, but only 53% of spammers are correctly classified, on average[1]. A further investigation revealed that several of these spammers are indeed very hard to identify based only on their behaviors as they act very similarly to legitimate users[2]. There is a strong assumption in the collection used in this work that may also contribute to the high amount of misclassified spammers. If a user posts a single polluted video, she is considered a spammer, even if she posts a lot of legitimate videos. Regarding the second baseline, Table 5.6 shows that it produces results that are in fact worse than the previous ones, mainly with regards to the classification of spammers. In particular, the fraction of correctly identified spammers (in the diagonal) dropped by more than 20%. Moreover, the fractions of promoters and of spammers that were misclassified as legitimate users increases by at least the same factor. As discussed in Section 4.2, such loss in classification performance is particularly worrisome as those users would probably escape undetected.

		Predicted		
		Promoter	Spammer	Legitimate
	Promoter	100%	0%	0%
True	Spammer	1.02%	53.25%	45.73%
	Legitimate	0%	0.78%	99.22%

Table 5.5: Classification with Baseline 1 (Supervised Method with 100% of Training Data)

[1] We should notice that these results are slightly different from those reported in Benevenuto et al. [2009a] because the classifiers used are different: we here use LAC, whereas the previous work used SVM.

[2] We found no clear distinction between the values of several attributes of such spammers and the typical values of the same attributes in legitimate users.

		Predicted		
		Promoter	Spammer	Legitimate
True	Promoter	96.13%	1.29%	2.58%
	Spammer	3.46%	41.92%	54.62%
	Legitimate	0%	2.57%	97.43%

Table 5.6: Classification with Baseline 2 (Supervised Method with 20% of Training Data)

5.3 Semi-supervised Results

Table 5.7 shows the confusion matrix obtained with the multi-view classification based on the View Agreement strategy. The problem with this approach is that the views do not agree with regard to the majority of spammers and promoters, as can be (partially) seen in Figures 5.1b) and 5.1c). Recall that the process of adding new samples to the training set stops once no agreement is obtained with respect to any class. Thus, this approach ends up adding only a small number of new promoters and spammers to the training set, limiting its effectiveness. When it comes to promoters, in spite of the small agreement rate, this approach is still able to add enough of them during all iterations of the algorithm so as to keep the initial proportions (see discussion in Section 3), because the number of promoters in the initial training set is very small. The real problem is with spammers. Figures 5.1b) and 5.1c) show that the agreement between the two views is even lower for this class. As consequence, the process of introducing new samples into the training set stops after only a few iterations of the algorithm (maximum of two, in our experiments), and very few new spammers (and users in general) are included in the training set. Thus, similarly to the baseline with 20% of the training set, this approach is very ineffective in the prediction of spammers. Moreover, semi-supervised approaches may add to the training set a few number of samples with incorrect classes. When the process continues for several iterations, this problem may become less prominent, if the majority of the insertions are correct. However, since here the number of iterations is very small, any sample inserted with the wrong class may significantly impact the classifier in its future decisions, ultimately leading it to wrong predictions. This is specially important in the case of promoters, as the number of elements of that class is very small.

Clearly, spammers is the most difficult class to predict, by both baselines and by the View Agreement approach. We should note that the much lower agreement between the two views with regard to that specific class may ultimately impact the multi-view classification of all three classes, as it causes the earlier interruption of the process of

		Predicted		
		Promoter	Spammer	Legitimate
True	Promoter	89.68%	7.10%	3.23%
	Spammer	3.21%	43.59%	53.21%
	Legitimate	0%	3.13%	96.87%

Table 5.7: Classification with the View Agreement Approach

adding new samples (of all classes) to the training set. As a matter of fact, we did observe in our experiments that the great disagreement between views for spammers is the main factor limiting the continuity of the training expansion process. In spite of that, we note that the results of that approach for promoters and legitimate users are reasonably good (Table 5.7), as they are based on enough agreement with higher confidence. Thus, we next explore a hybrid approach that applies the Borda Count algorithm, described in Section 3.2, only for spammers, keeping the View Agreement strategy for the other 2 classes. In other words, we create the candidate sets of legitimate users and promoters according to the agreement between the two views (as in Algorithm 1) and the candidate set of spammers according to the final aggregated ranking produced by Algorithm 2. We then select users from each set according to the corresponding criteria (confidence for legitimate users and promoters, and rank for spammers), keeping the relative proportions similar as in the initial training set.

Table 5.8 shows the confusion matrix with the results of this hybrid approach. In comparison with the baseline with 100% of training, this approach is only slightly worse in predicting promoters and legitimate users. However it achieves comparable (and, in some folds, slightly better) performance when it comes to correctly identifying spammers, the hardest class. These results are indeed quite promising considering the great reduction (by a factor of 5) on the required amount of labeled data. Moreover, given the envisioned application of our technique as a tool to help system administrators by filtering suspicious users for further (manual) investigation, we believe the results for promoters and legitimate users are also quite positive. The small fraction of misclassified promoters were considered as spammers, that is, they were predicted at least as non-cooperative users. Moreover, the fraction of misclassified legitimate users is reasonably small. The misclassification of such users could be reversed during manual investigation. Specifically for the collection used in this work, the reduction in the required amount of labeled data means that 530 less users need to be manually evaluated, while the increase of misclassified legitimate users means that only 10 additional legitimate users (7.37% of them) need to be manually evaluated after the classification process. In very large user collections, this tradeoff needs to be better studied.

However, it is always worth to remind the various costs associated with the action of spammers which justify a more aggressive approach towards identifying them. There are direct costs associated with the use of bandwidth, network, and cache, and indirect costs due to a possible loss of credibility and reputation of the service from the dissatisfaction of the users, which are perhaps even worse than the direct costs. Because of this, a small increase in the number of users that have to be manually inspected later is acceptable, provided there is an increase in the amount of spammers being identified correctly. In comparison with the baseline with 20% of training, our approach is able to improve spammer detection by 35%, at the cost of only a slight degradation (6%) in the correct classification of legitimate users. We should also note that, in comparison with the basic View Agreement approach, the hybrid strategy improved the correct classification of both classes of non-cooperative users, indicating that, indeed, the very low agreement of both views with regard to spammers impacted the classification of both classes.

		Predicted		
		Promoter	Spammer	Legitimate
True	Promoter	96.77%	3.23%	0%
	Spammer	3.21%	56.67%	40.13%
	Legitimate	0.09%	8.15%	91.76%

Table 5.8: Classification with the Hybrid Borda Count / View Agreement Approach

For sake of completeness, we also experimented with the Borda Count approach applied to all three classes, as can be seen in Table 5.9. The use of Borda Count approach for the three classes has proved to be worse than the hybrid approach used before, erroneously classifying 27% of promoters and 57% of spammers. The Borda count uses each view confidence individually in the sense that it first produces a ranking of users for each view before the rank aggregation takes place. Thus, misclassifications with high confidence (e.g., spammers classified as legitimates with high confidence) may cause wrong insertions into the training data that can be avoided when the insertions are based on the high levels of agreement that occur specifically for the legitimate and promoters classes, as observed in our datasets. Thus, the additional spammers and promoters incorrectly added to the training set when compared to the view agreement approach, caused an overall drop in the classification effectiveness.

		Predicted		
		Promoter	Spammer	Legitimate
True	Promoter	72.90%	16.13%	10.97%
	Spammer	2.18%	42.95%	54.87%
	Legitimate	0%	4.08%	95.92%

Table 5.9: Classification with the Borda Count Approach

Finally, we also analyze the performance of the methods under the macro-F1 metric. The results for the supervised method with 100% and 20% of training data are 0.86 and 0.78, respectively. In comparison, the macro-F1 results for the View Agreement and the Hybrid approaches are, respectively, 0.77 and 0.8. Thus, in terms of this metric, the multi-view approach combining View Agreement and Borda Count strategies is slightly better than the other approaches using the same amount of training (20%), and it is only around 7% worse than the classifier with 5 times more training data. Nevertheless, if the correct classification of non-cooperative users is favored at the expense of misclassifying legitimate users, the results in Tables 5.5-5.8 show much more clearly the superiority of the proposed Hybrid View Agreement / Borda Count approach.

Chapter 6

Analyzing the Costs of Detecting Non-cooperative Users

Up to this point, we have analyzed the non-cooperative user detection methods (i.e., our new methods and the baseline) in terms of classification effectiveness, i.e., our focus was on the quality of classification results. However, the use of each method incurs in costs to the system administrator. Such costs may vary depending on the method employed as well as on the user collection on which it is applied. Thus, a method that has a good effectiveness is not necessarily the most suitable, because it may incur in a very high utilization cost. To the best of our knowledge, there is no previous analysis nor study of such costs. Moreover, we are not aware of publicly available reports on actual costs related to the manual classification of videos and of their users nor on the costs associated with the polluted videos that remain undetected and thus are stored by real systems. Thus, in this chapter, we develop an analytical model to estimate the costs associated with the utilization of different methods to identify non-cooperative users in OVSSs. We here apply this model in different scenarios in order to compare two methods: our best proposed method (the hybrid method) and the baseline that uses all the training data available. Next, we first introduce our analytical cost model (Section 6.1) and then use it to evaluate the costs associated with the two methods (Section 6.2).

6.1 Analytical Cost Model

We envision three different cost components associated with the deployment and utilization of a mechanism to detect non-cooperative users: (1) the training cost, (2) the cost associated with the non-cooperative users which are not caught by the method

(false negatives) and (3) the cost (if any) associated with the users predicted as non-cooperative. We discuss each cost component next.

If a supervised detection method, such as the ones analyzed in this work, is used, each video owned by the users in the training set must be viewed and manually classified as either spam or not. As result of this labeling, the users themselves must be manually classified as either spammer, promoter or legitimate user. This manual classification incurs in a cost to the system administrator, which is here referred to as the *training cost*. A second cost component relates to the non-cooperative users who are not caught by the method. Such false negatives incurs in costs to the system administrator in several ways. As mentioned in Chapter 5, there are direct costs associated with the use of storage and network resources to deliver the spams posted by such users to other users when they inadvertedly request them. There are also indirect costs related to a possible loss of credibility and reputation of the service caused by user dissatisfaction. Such indirect costs are perhaps even more significant and certainly harder to estimate. Finally, the third cost component is associated with the users that are classified as non-cooperative by the method. As mentioned in Chapter 5, the idea is that system administrators will use the techniques showed in this work to filter suspicious users for further (manual) investigation. Therefore, this pos-processing does generates extra costs which are directly proportional to the number of users that the method classifies as non-cooperative.

We now introduce an analytical model developed to capture the three aforementioned cost components. Table 6.1 summarizes all the variables used in our model.

Notation	Definition
c_{Eval}	Cost to evaluate a single video
$c_{Pollution}$	Cost associated with a single polluted video left in the system
n_{Train}	Number of users in the training set
n_{Test}	Number of users in the test set
n_{NC}	Number of users classified as non-cooperative
$n_{FalseLeg}$	Number of non-cooperative users classified as legitimate
v_{NC}	Average number of videos posted by each non-cooperative user
v_{User}	Average number of videos posted by a user

Table 6.1: Variables involved in the cost model

The training cost depends on three factors, namely, the cost of evaluating a single video (c_{Eval}), the average number of videos posted by a user (v_{User}) and the number of users in the training set (n_{Train}). Thus, the training cost is defined as the product of these three factors, that is:

$$c_{Train} = c_{Eval} * v_{User} * n_{Train} \tag{6.1}$$

The cost associated with the false negatives depends on three factors, namely, the cost associated with a single polluted video left in the system ($c_{Pollution}$), the average number of videos posted by each non-cooperative user (v_{NC}) and the number of non-cooperative users classified as legitimate ($n_{FalseLeg}$). The cost associated with false negatives is thus defined as the product of these three variables, being defined as:

$$c_{FN} = c_{Pollution} * v_{NC} * n_{FalseLeg} \tag{6.2}$$

The cost associated with users classified as non-cooperative depends on three factors, namely, the cost to manually evaluate a single video (c_{Eval}), the average number of videos posted by a user (v_{User}) and the number of users classified as non-cooperative (n_{NC}). The cost to manually evaluate all videos of users classified as non-cooperative is thus given by:

$$c_{NC} = c_{Eval} * v_{User} * n_{NC} \tag{6.3}$$

Given the definition of each cost component (Equations 6.1, 6.2 and 6.3), the total cost associated with the utilization of a given method is defined as:

$$cost = c_{Train} + c_{FN} + c_{NC} \tag{6.4}$$

Rearranging the variables, we can simplify the model and write it as:

$$cost = c_{Eval} * v_{User} * (n_{Train} + n_{NC}) + c_{Pollution} * v_{NC} * n_{FalseLeg} \tag{6.5}$$

We note that the values of almost all variables on which our model relies, defined in Table 6.1, depend on the specific detection method and user collection used. For example, methods that produce different classification results, i.e., that classify a different number of users as being of each class lead to different values of variables $n_{FalseLeg}$ and n_{NC}. Moreover, methods that uses different amounts of training impact on variable n_{Train}. Also, different user collections impact variables v_{NC} and v_{User}, as the users in different collections may differ in terms of number of videos posted. The cost variables, c_{Eval} e $c_{Pollution}$, on the contrary, do not depend neither on the method nor on the user collection, but rather on several other aspects. For example, c_{Eval} captures the cost to manually evaluate a single video, which is hard to estimate, because we do not know how much money an OVSS would pay to a person to watch and

classify a single video. Also, $c_{Pollution}$ captures the costs associated with the storage and bandwidth resources wasted by the system with polluted content in addition to other indirect costs associated with loss of system reputation. As we are not aware of publicly available estimates of absolute values (e.g., in dollars) of either cost variable, we here assume such costs are defined in terms of a pre-defined cost unit u.

In sum, given the values of the input variables in Table 6.1, parameterized to define specific scenarios of interest and to address a given detection mechanism, our model can be used to estimate the total cost (in the given cost unit u) associated with the utilization of that method. In the next section, we design different scenarios by parameterizing our model based on statistics extracted from our user collection (presented in Chapter 4) and based on the classification results produced by each of the two analyzed methods, which were discussed in Chapter 5.

6.2 Cost Analyses

In this section, we use the analytical model to analyze and compare the costs associated with two methods: the baseline, which uses all available training data, and our best proposed method, i.e., the hybrid Borda Count / View Agreement approach. This analysis is performed over the same user collection used in the experiments presented in Chapter 5, which was introduced in Chapter 4.

As mentioned in the previous section, almost all the variables shown in Table 6.1 are determined by the collection itself and by the detection method used, with the exception of variables c_{Eval} and $c_{Pollution}$. Given our selected user collection, Table 6.2 shows the values of each of the variables for the two analyzed detection methods. As we can see, three variables, namely v_{NC}, v_{User}, and n_{Test}, depend only on the collection, and thus have the same values regardless of the method used. The values of the other three variables are defined based on the specific detection method used.

Variable	Value for Baseline	Value for Hybrid Method
n_{Train}	16,600	3,317
n_{Test}	4,125	4,125
n_{NC}	604	885
$n_{FalseLeg}$	357	313
v_{NC}	13.83	13.83
v_{User}	24.64	24.64

Table 6.2: Values of Most Variables of Analytical Model for the given Collection and Method

To perform our analyses, we create different scenarios, based on the input variables of the analytical model. We then analyze the total costs associated with each method in each scenario. In each scenario we will use some ratios between the variables involved in the cost model. These ratios are shown in Table 6.3. The first set of scenarios is designed to analyze the impact of the ratio $C = c_{Pollution}/c_{Eval}$ on the total cost associated with each method. This ratio represents the relative cost of manual evaluation of a single video with respect to the cost of keeping a single polluted video in the system. Thus, we use the values of each variable in Table 6.2 to estimate the total cost associated with each method, for various values of C. The results are shown in Figure 6.1.

Ratio	Definition
C	$C = c_{Pollution}/c_{Eval}$
V	$V = v_{NC}/v_{User}$
F	$F = n_{FalseLeg}/n_{NC}$

Table 6.3: Ratios Used in the Analyzed Scenarios

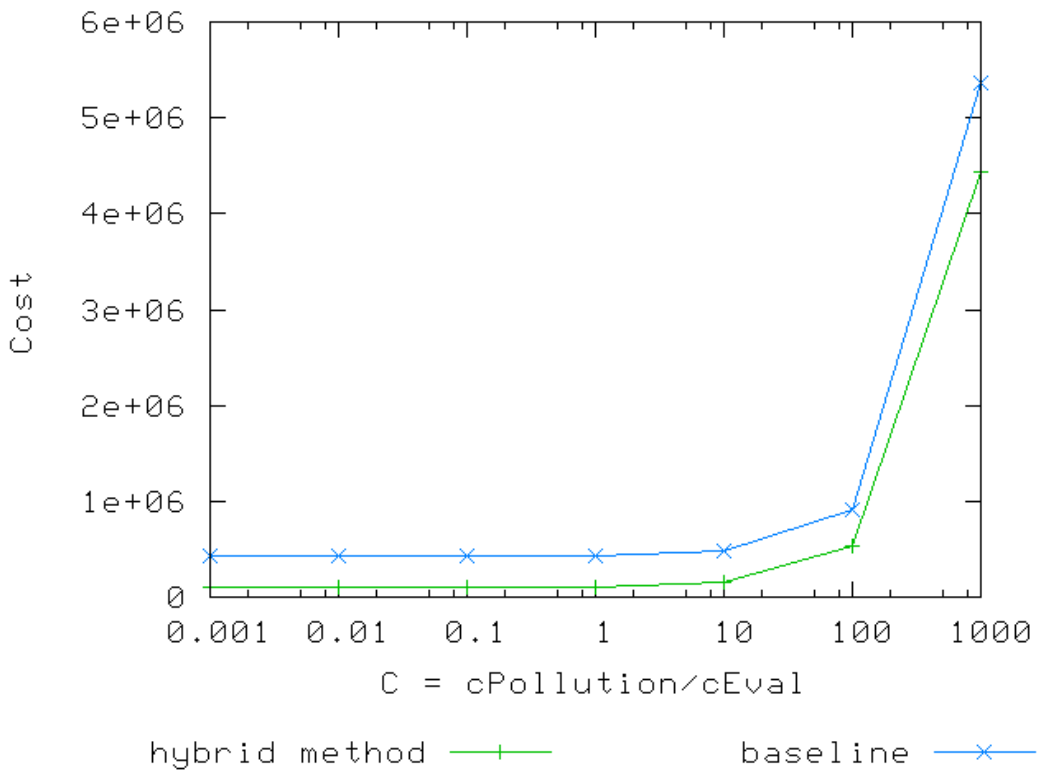


Figure 6.1: Total Costs as Functions of $C = c_{Pollution}/c_{Eval}$

For small values of C, i.e., when $cPollution$ is much smaller than c_{Eval}, the training cost dominates the total cost associated with each method (Equation 6.5).

Therefore, the baseline has a higher cost in this case, as it uses a much larger training set. As C increases, the cost associated with non-cooperative users that are not caught by the method becomes more significant, and starts dominating the total costs. Again, in this case, the baseline also has a higher total cost than the hybrid method as, according to our discussion in Chapter 5, it lets more non-cooperative users pass undetected, thus having a larger number of false negatives (i.e., higher value of $n_{FalseLeg}$). Indeed, as shown in Figure 6.1, the hybrid approach always incurs in a lower total cost than the baseline. For example, for $C = 0.01$, the baseline incurs in a total cost that is 309% larger than the cost associated with the hybrid method. Similarly, for $C = 100$ and $C = 1000$, the total costs associated with the baseline are 71% and 21% larger than the costs of our hybrid method, respectively. In sum, considering the scenarios built using the variable values defined in Table 6.2, the baseline always incurs in more costs to the system administrator than the hybrid method independent of the ratio $c_{Pollution}/c_{Eval}$.

Another factor that might impact the costs of each method is the size of the test set (n_{Test}), as it affects the number of users classified as non-cooperative (n_{NC}) and the number of non-cooperative users classified as legitimate ($n_{FalseLeg}$). We now analyze the total costs of each method varying the size of the test set. To do so, we fix $C = 1$, and define a new variable T which indicates the size of the test set as a multiple of the size of the test set in our user collection (i.e., $T = 1$, implies $n_{Test} = 4,125$, whereas $T = 2$, implies $n_{Test} = 8,250$). For this analysis, we assume test sets that, despite larger, keep the same fractions of users from each class as in the test set of our user collection, and, thus, that each method produces similar classification results (percent-wise) as observed in our original collection (Chapter 5). This assumption of the maintainance of the relative distribution of the classes' proportions is also assumed by most machine learning algorithms in order to guarantee the generalization of results. The results are shown in Figure 6.2.

For small values of T, the hybrid method has a lower total cost in comparison with the baseline because the training cost dominates the total cost (Equation 6.5). As T increases, the cost associated with the number of users classified as non-cooperative becomes more significant. In this case, the hybrid method has a higher total cost than the baseline, because, according to the results discussed in Chapter 5, it classifies more users as spammers, i.e., has a higher value of n_{NC}. Thus, the two curves intersect when the test set is 52 times larger than it is in the collection we are using. This means that if we consider the variables c_{Eval} and $c_{Pollution}$ having the same value, the baseline incurs in lower total cost than the hybrid method when the test set has at least 214,500 users. This value may vary depending of the value of the C variable.

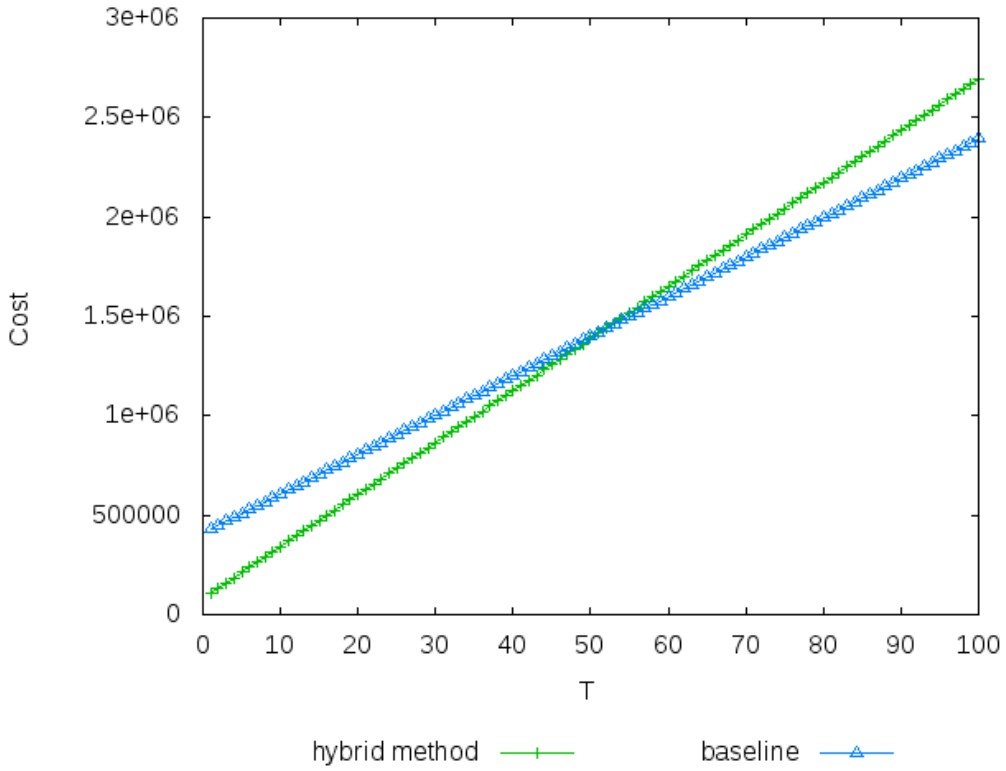


Figure 6.2: Total Costs as Functions of T (size of test set as multiple of test set in our collection)

We turn back to our analysis of the cost ratio C, considering now a test set 100 times larger than the one in the collection used. The results are shown in Figure 6.3. Note that both X and Y axes are in logarithmic scales. As expected, in this case, the hybrid method is not always better than the baseline. For small values of C, the cost associated with the users classified as non-cooperative dominates the total cost. The hybrid method has a somewhat higher cost than the baseline, because it predicts more users as non-cooperative. For example, for $C = 0.01$ the hybrid method has a total cost that is 19% higher than the costs associated with the baseline. As C increases, the cost associated with false negatives becomes the dominant cost component. As consequence, the baseline incurs in a higher cost than the hybrid method, because it leads to more false negatives, i.e., has a higher value of $n_{FalseLeg}$. For example, for $C = 100$ the baseline has a total cost that is 13% higher than the costs associated with the hybrid method.

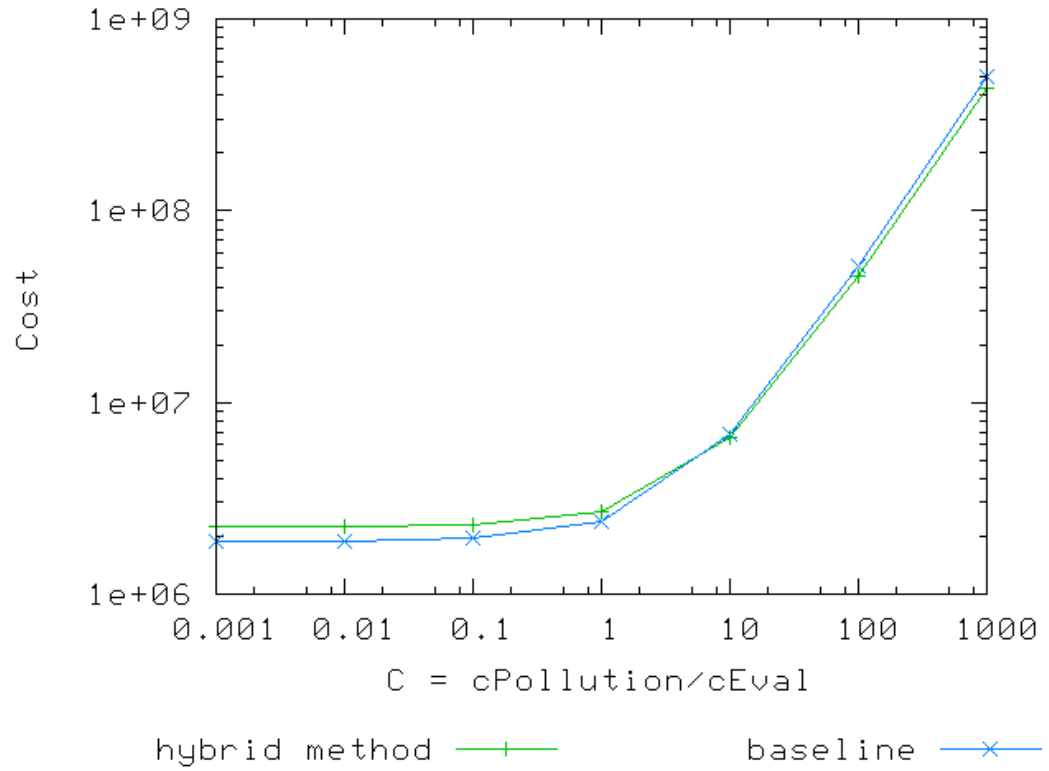


Figure 6.3: Total Costs as Functions of $C = c_{Pollution}/c_{Eval}$ for T = 100

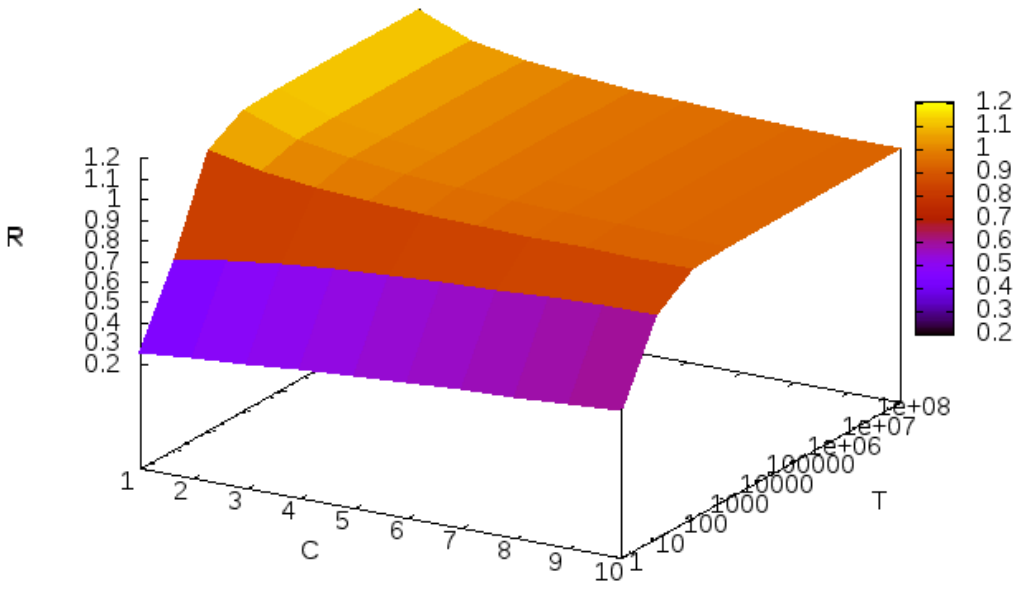


Figure 6.4: Total Costs as Functions of C and T variables

We summarize the impact of variables C and T on the total costs associated with the methods by plotting, in Figure 6.4, the ratio R of the total cost associated with the hybrid method to the total cost associated with the baseline for various values of C and T. Note that values of R lower than 1 imply in a lower total cost for our method whereas values of R greater than 1 imply in a lower cost for the baseline. As we can see in the Figure, for values of C equal to or larger than 4, i.e., $c_{Pollution}$ is at least 4 times larger than c_{Eval}, the hybrid method has a lower total cost than the baseline for a test set up to 100 million times bigger than it is in the collection used, i.e., for a test set having up to 412.5 billion users. This value of C is plausible, since, in practice, it is expected that $c_{Pollution}$ would have higher values than c_{Eval} in most situations, because of the inumerous factors involved in the value of $c_{Pollution}$, as discussed in Section 6.1.

Now we will analyze the total costs associated with each method in two sets of scenarios. First, we analyze the costs as we vary $V = v_{NC}/v_{User}$, i.e., the ratio of the average number of videos posted by each non-cooperative user to the average number of videos posted by a user. Afterwards, we analyze the cost of each method as we vary the ratio $F = n_{FalseLeg}/n_{NC}$, that is, the ratio of the number of non-cooperative users that are not detected to the number of users classified as non-cooperative. To compute the costs, we first need to choose a value for the cost ratio C. Since this value is only a multiplicative factor in the computation of the cost, different values of C will only make the curves intersect at different points in the graph, but will not change the general behavior of the curves. Thus, we choose to use the value $C = 10$, since in practice it is expected $c_{Pollution}$ to be bigger than c_{Eval}.

We start analyzing the costs of each method as a function of V (Table 6.3. As we did for C, we analyze the costs for two sizes of test sets, namely, $T = 1$ (size equal to the test set in our collection) and $T = 100$. Figure 6.5 shows the total costs of each method as functions of V, fixing a test set with size $T = 1$. If we check the cost model (Equation 6.5), we can see that when we consider c_{Eval} and v_{User} both equal to 1 and vary the value of v_{NC} (which is exactly what we do in this scenario) the cost model has a behavior similar to when we vary the C variable. In other words, for small values of V, the training cost dominates the total cost, and thus the hybrid method has a lower total cost than the baseline. As V increases, the false negatives starts to dominate the total cost, which again makes the hybrid method have a lower total cost than the baseline. Thus, the curves in Figure 6.5 are expected to have a similar behavior to the ones in Figure 6.1, i.e., the hybrid method is expected to always have a lower total cost than the baseline. For example, for $V = 0.01$, the baseline has a total cost that is 307% higher than the cost associated with the hybrid method. Similarly, for $V = 100$ the baseline has a total cost that is 18% higher than the cost associated with the

hybrid method. The values of V may vary in different collections, since the users may have different behaviors in each OVSS. Particularly, in our collection $V = 0.56$ meaning that legitimate users post almost twice more videos when compared to non-cooperative users, as we can see in Table 6.2.

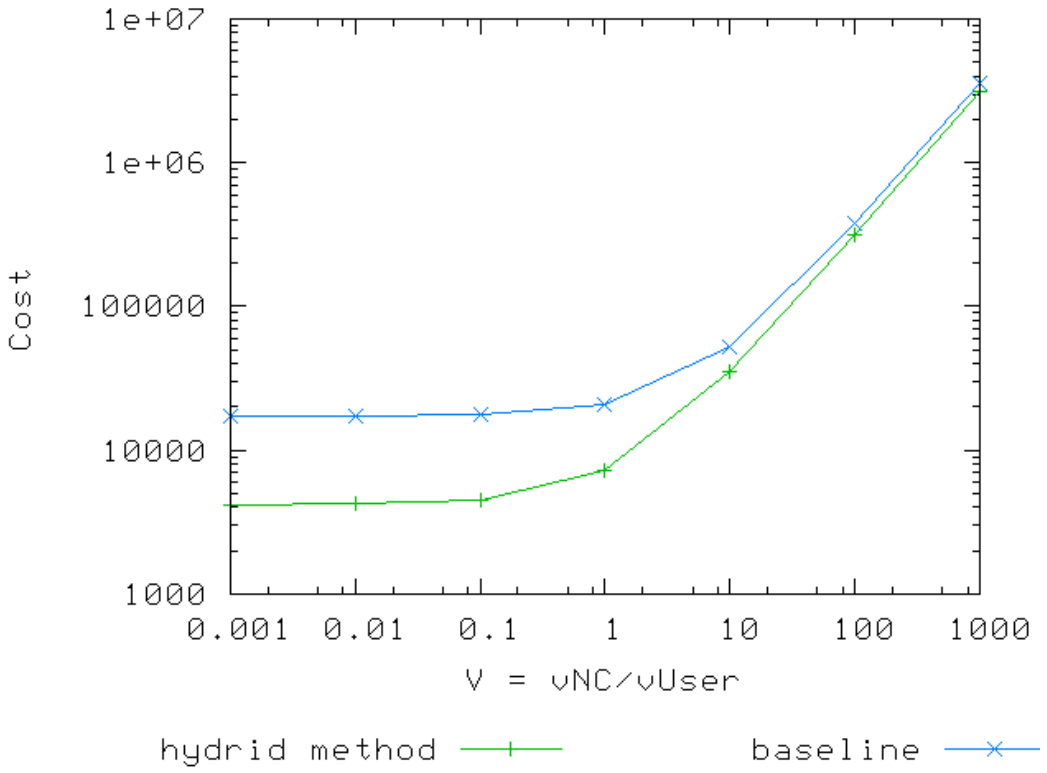


Figure 6.5: Total Costs as Functions of $V = v_{NC}/v_{User}$ for T = 1

The results for $T = 100$, shown in Figure 6.6, exhibit similar patterns to those obtained as function of C for the same size of test set (Figure 6.3). In other words, for small values of V, the cost associated with the users classified as non-cooperative dominates the total cost. Thus the hybrid method has a higher total cost than the baseline. As V increases, the cost associated with the false negatives starts to dominate the total cost. As consequence, the total cost associated with the baseline increases, becoming larger than the costs of the hybrid method. Indeed, Figure 6 shows that the baseline incurs in a lower total cost than the hybrid method for values of V smaller than 0.25, and the hybrid method incurs in a lower total cost than the baseline for values of V larger than 0.75. To ilustrate this, for $V = 0.01$, the hybrid method has a total cost that is 18% higher than the cost associated with the baseline, and for $V = 100$ the baseline has a total cost that is 14% higher than the cost associated with the hybrid method.

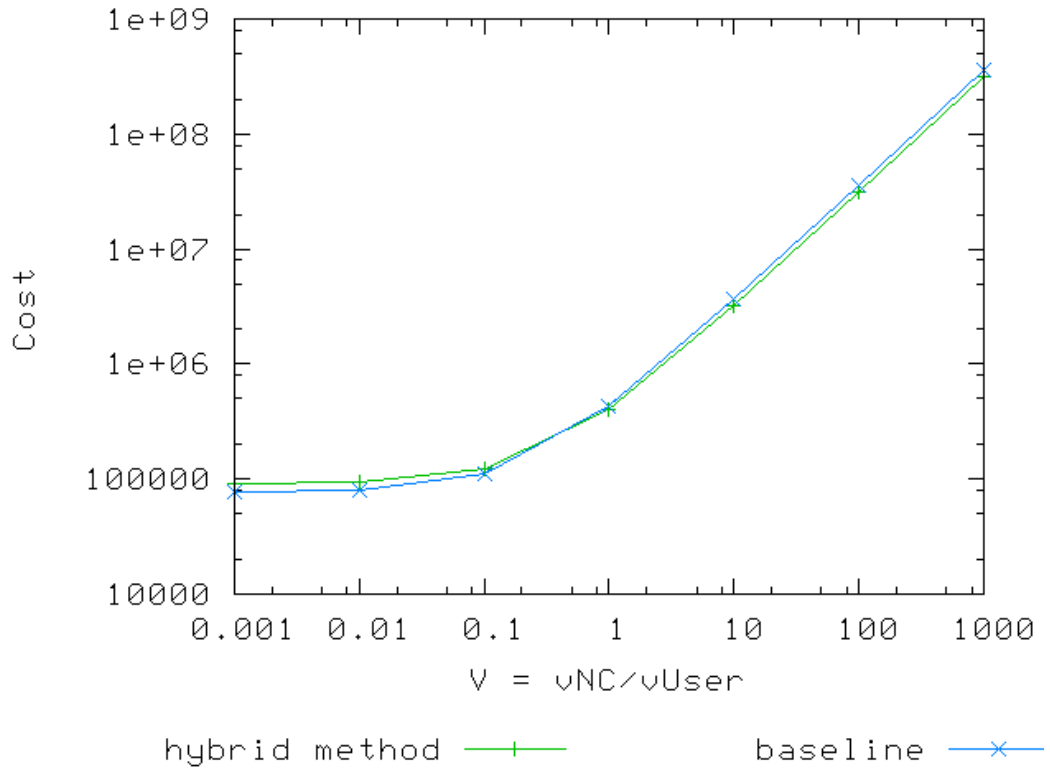


Figure 6.6: Total Costs as Functions of $V = v_{NC}/v_{User}$ for T = 100

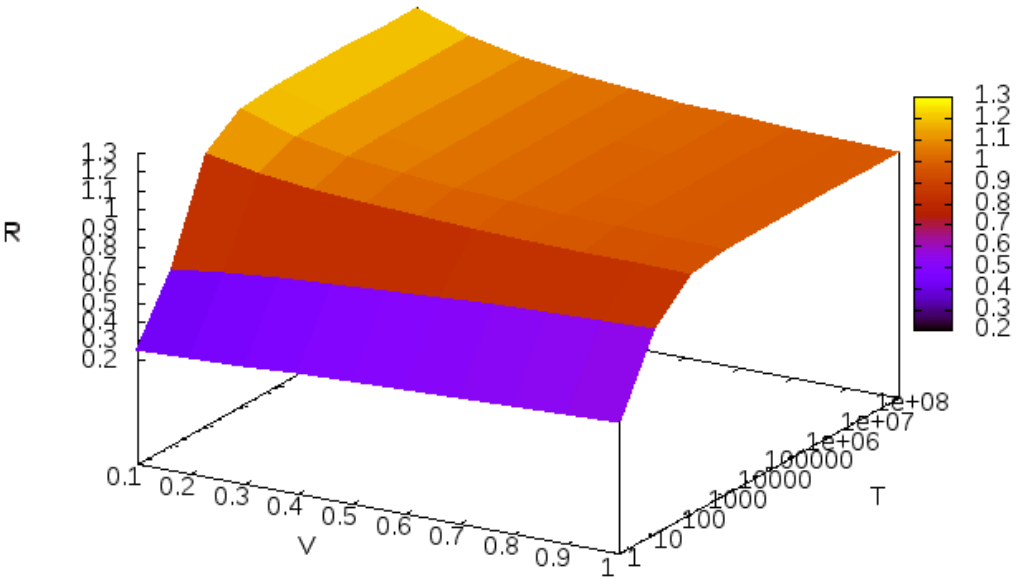


Figure 6.7: Total Costs as Functions of V and T

Figure 6.7 shows the ratio R of the total cost associated with the hybrid method to the total cost associated with the baseline for various values of V and T. Once again, values of R lower than 1 imply in a lower total cost for our method whereas values of R greater than 1 imply in a lower cost for the baseline. As we can see in the figure, for values of V equal to or larger than 0.7, the hybrid method has a lower total cost for test sets with sizes up to 100 million times bigger than the size of the test set in the collection used, i.e., for a test set having up to 412.5 billion users. This means that when the average number of videos posted by each non-cooperative user is equal, greater or slightly lower than the average number of videos posted by a user, the hybrid method will have a lower cost than the baseline for the aforementioned sizes of the test sets.

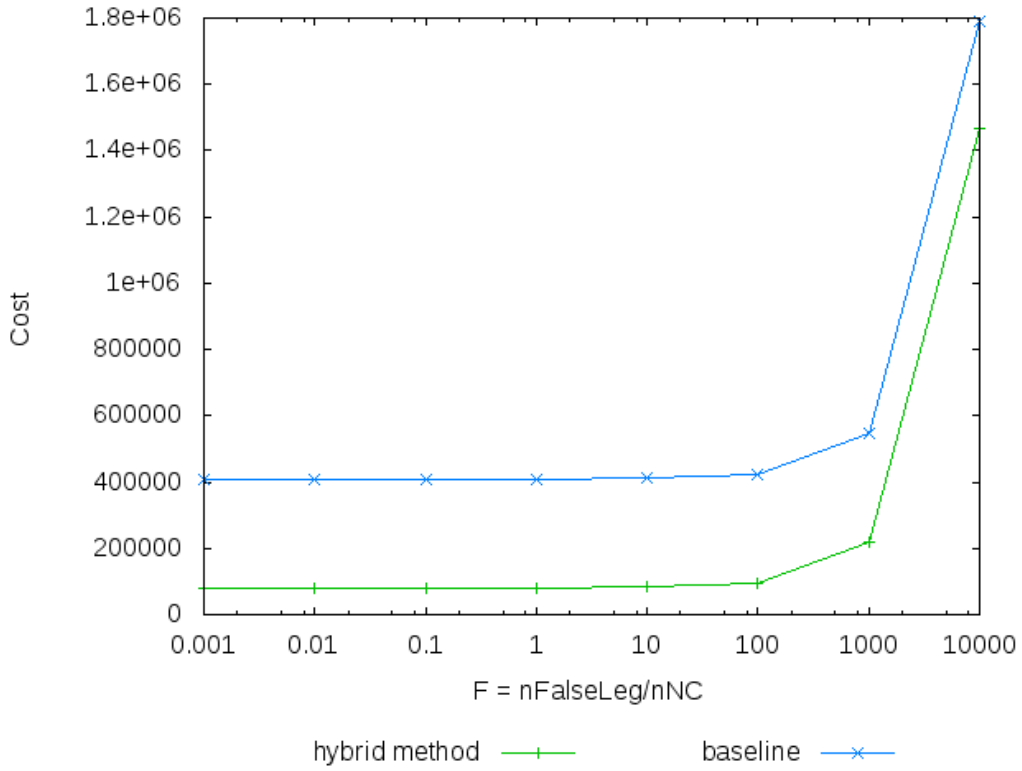


Figure 6.8: Total Costs as Functions of $F = n_{FalseLeg}/n_{NC}$

As a last set of scenarios, we analyze the costs of each method as function of the ratio $F = n_{FalseLeg}/n_{NC}$ (Table 6.3), that is, the ratio of the number of non-cooperative users that are not detected to the number of users classified as non-cooperative. In this case, different sizes of the tests set do not change the total cost. This is because we assume that both methods produce similar results (percent-wise) for different sizes of test sets. In other words, we assume that even though the absolute values of $n_{FalseLeg}$

and n_{NC} change as the size of the test set increases, their ratio F remains fixed. Therefore, Figure 6.8 shows the costs obtained as functions of F. Note that, for small values of F, the training cost dominates the total cost, leading to a higher cost for the baseline. For example, for $F = 0.01$ the baseline has a total cost that is 400% higher than the costs associated with the hybrid method. As F increases, the cost associated with false negatives starts dominating the total cost, and once again, the total costs associated with the baseline is higher. For example, for $F = 1,000$ the baseline has a total cost that is 149% higher than the costs associated with the hybrid method.

In sum, based on our analyses, we conclude that, for test sets much larger than the one we have in the collection used, the hybrid method is a good choice, obtaining lower costs than the baseline in a good portion of the scenarios analyzed. In particular, focusing on the impact of C and T and keeping the other variables fixed at the values extracted from our collection, we find that if C has a value larger than or equal to 4, the hybrid method has a lower cost than the baseline for test sets up to 100 million times larger than the one we used. This value of the C variable is plausible, since it is expected that $c_{Pollution}$ has a higher value than c_{Eval} and, sometimes, a much higher value.

Chapter 7

Conclusions and Future Work

Our proposed method explores a multi-view semi-supervised classification algorithm, which requires a much smaller amount of training data than previously proposed supervised methods. We evaluate two approaches, built from two different strategies for combining results from multiple views, using a sample of pre-classified users and a set of user behavior attributes. In comparison with supervised methods, our best approach, which combines the View Agreement and the Borda Count view combination strategies, achieves a much more favorable tradeoff between detecting non-cooperative users and reducing amount of training data, at the possible expense of a slight increase in the fraction of misclassified legitimate users. In particular, it achieves comparable performance, particularly on detecting spammers (the hardest class), with 5 times fewer training data than the supervised method.

Furthermore, we developed an analytical cost model to compare our best method with the baseline used. The analysis of this cost model in different scenarios showed that our proposed approach has a lower cost than the baseline in the majority of the cases. An interesting result is that when the cost of not identifying a video pollution in the system is at least four times greater than the cost of manual evaluation of a video, our proposed approach has a lower cost than the baseline for testing sets having up to 412.5 billion users.

As future work, we intend to build a new, much larger, user test collection as well as explore the use of a fourth view - the video content view - as means to further improve spammer detection, by more clearly distinguishing them from legitimate users. We also plan to test other strategies to combine views and test other hybrid approaches, combining these new strategies with the ones we used in this work. Finally, we want to integrate our methods with active learning methods applied to each view. In other words, we want to use an active learning approach to select, from each view, wich

elements will be added to the training set. This could reduce even further the need for training.

Bibliography

Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *SIGMOD*, pages 207--216.

Benevenuto, F., Rodrigues, T., Almeida, V., Almeida, J., and Gonçalves, M. (2009a). Detecting spammers and content promoters in online video social networks. In *SIGIR*.

Benevenuto, F., Rodrigues, T., Almeida, V., Almeida, J., and Ross, K. (2009b). Video interactions in online video social networks. *ACM Trans. Multimedia Comput. Commun. Appl.*, 5(4):1--25.

Black, D. (1958, 1963). *The theory of committees and elections*. Cambridge University Press, London :, 2nd eds. edition.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *COLT*, pages 92--100.

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107--117.

Castillo, C., Donato, D., Murdock, V., and Silvestri, F. (2007). Know your neighbors: Web spam detection using the web topology. In *SIGIR*.

Christoudias, C., Urtasun, R., and Darrell, T. (2008). Multi-view learning in the presence of view disagreement. In *UAI*.

comScore (2010a). comscore releases march 2010 u.s. online video rankings. http://www.comscore.com/Press_Events/Press_Releases/2010/4/comScore_Releases_March_2010_U.S._Online_Video_Rankings.

comScore (2010b). comscore releases march 2010 u.s. search engine rankings. http://www.comscore.com/Press_Events/Press_Releases/2010/4/comScore_Releases_March_2010_U.S._Search_Engine_Rankings.

Dwork, C., Kumar, R., Naor, M., and Sivakumar, D. (2001). Rank aggregation methods for the web. In *WWW*, pages 613--622.

Fetterly, D., Manasse, M., and Najork, M. (2004). Spam, damn spam, and statistics: Using statistical analysis to locate spam web pages. In *WebDB*, pages 1--6.

Gao, H., Hu, J., Wilson, C., Li, Z., Chen, Y., and Zhao, B. Y. (2010). Detecting and characterizing social spam campaigns. In *Proceedings of the 10th annual conference on Internet measurement*, IMC '10, pages 35--47, New York, NY, USA. ACM.

Gomes, L. H., Cazita, C., Almeida, J. M., Almeida, V., and Meira, Jr., W. (2007). Workload models of spam and legitimate e-mails. *Perform. Eval.*, 64(7-8):690--714.

Gyöngyi, Z., Garcia-Molina, H., and Pedersen, J. (2004). Combating web spam with trustrank. In *VLDB*, pages 576--587.

Heymann, P., Koutrika, G., and Garcia-Molina, H. (2007). Fighting spam on social web sites: A survey of approaches and future challenges. *IEEE Internet Computing*, 11(6):36–45.

Hovelynck, M. and Chidlovskii, B. (2010). Multi-modality in one-class classification. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 441--450, New York, NY, USA. ACM.

Kohavi, R. and Provost, F. (1998). Glossary of terms. *Special issue on applications of machine learning and the knowledge discovery process*, 30(2-3):271--274.

Koutrika, G., Effendi, F. A., Gyöngyi, Z., Heymann, P., and Garcia-Molina, H. (2007). Combating spam in tagging systems. In *AIRWeb*, pages 57--64.

Lee, K., Caverlee, J., and Webb, S. (2010). Uncovering social spammers: social honeypots + machine learning. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 435--442, New York, NY, USA. ACM.

Lim, E.-P., Nguyen, V.-A., Jindal, N., Liu, B., and Lauw, H. W. (2010). Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 939--948, New York, NY, USA. ACM.

Lin, Y.-R., Sundaram, H., Chi, Y., Tatemura, J., and Tseng, B. L. (2008). Detecting splogs via temporal dynamics using self-similarity analysis. *ACM Trans. Web*, 2(1):1--35.

Newman, M. and Park, J. (2003). Why social networks are different from other types of networks. *Physical Review E*, 68(3):36122.

Perez, C. A., Cament, L. A., and Castillo, L. E. (2011). Methodological improvement on local gabor face recognition based on feature selection and enhanced borda count. *Pattern Recogn.*, 44:951--963.

Thomason, A. (2007). Blog spam: A review. In *CEAS*.

Veloso, A., Meira Jr., W., and Zaki, M. J. (2006). Lazy associative classification. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pages 645--654, Washington, DC, USA. IEEE Computer Society.

Wang, A. H. (2010). Detecting spam bots in online social networking sites: a machine learning approach. In *Proceedings of the 24th annual IFIP WG 11.3 working conference on Data and applications security and privacy*, DBSec'10, pages 335--342, Berlin, Heidelberg. Springer-Verlag.

Wu, C.-T., Cheng, K.-T., Zhu, Q., and Wu, Y.-L. (2005). Using visual features for anti-spam filtering. In *ICIP*, pages 509–512.

Xie, Y., Yu, F., Achan, K., Panigrahy, R., Hulten, G., and Osipkov, I. (2008). Spamming botnets: signatures and characteristics. In Bahl, V., Wetherall, D., Savage, S., and Stoica, I., editors, *SIGCOMM*, pages 171–182.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1-2):69--90.

YouTube (2010). Youtube fact sheet. http://www.youtube.com/t/fact_sheet.