Filtering Offensive Language in Online Communities using Grammatical Relations

Zhi Xu
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
zux103@cse.psu.edu

Sencun Zhu
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
szhu@cse.psu.edu

ABSTRACT

Offensive language has arisen to be a big issue to the health of both online communities and their users. To the online community, the spread of offensive language undermines its reputation, drives users away, and even directly affects its growth. To users, viewing offensive language brings negative influence to their mental health, especially for children and youth.

When offensive language is detected in a user message, a problem arises about how the offensive language should be removed, i.e. the *offensive language filtering problem*. To solve this problem, manual filtering approach is known to produce the best filtering result. However, manual filtering is costly in time and labor thus can not be widely applied.

In this paper, we analyze the offensive language in text messages posted in online communities, and propose a new automatic sentence-level filtering approach that is able to semantically remove the offensive language by utilizing the grammatical relations among words. Comparing with existing automatic filtering approaches, the proposed filtering approach provides filtering results much closer to manual filtering.

To demonstrate our work, we created a dataset by manually filtering over 11,000 text comments from the YouTube website. Experiments on this dataset show over 90% agreement in filtered results between the proposed approach and manual filtering approach. Moreover, we show the overhead of applying proposed approach to user comments filtering is reasonable, making it practical to be adopted in real life applications.

1. INTRODUCTION

Online social networking (OSN) websites have enjoyed a great success in recent years. People in OSN websites form social aggregations, called online communities [8]. These online communities have become the new frontier in today's social relationships and provide great places for self-expression and the exchange of ideas. Many of them, such as Facebook, have grown to huge communities with millions of registered members[1].

Online communities, being virtual, however, have also encouraged the use of offensive language. The definition of offensive language can be subjective because different viewers have different feelings about the same content. In this paper, we accept the definition of offensive language as text content including gutter language, sexually explicit material, racist, graphic violence, or any other content that may be considered offensive on social, religious, cultural or moral grounds[2].

Unfortunately, offensive language has spread into almost every corner of online communities. A study, done by ScanSafe, shows that up to 80% of blogs contain offensive language [5]. Posting messages with offensive language intentionally has become a major way of cyber-bullying in online communities. To users, offensive language can be very harmful to their mental health, especially for children and youth. To the online community, the deluge of offensive language undermines the community's reputation, drives users away, and even directly affects its growth. For example, an iPhone application, Tweetie, was once rejected by Apple company in March 2009, for bringing offensive language posted in the Twitter community to iPhone users.

People have realized the problems brought by offensive language in online communities. And many efforts have been made on detecting the existence of offensive language within user messages, such as [10] and [3].

However, detection alone is not enough to eliminate the hazard caused by offensive language. When offensive contents (e.g., offensive words) are detected within a user message, a question arises naturally about how the detected offensive content should be removed from message. The process of removing offensive content from user messages is called *offensive language filtering*. Consider a sentence consisting of a sequence of words. The problem of identifying words that should be removed in offensive language filtering is called *offensive language filtering problem*.

In this paper, we propose a sentence-level semantic filtering approach, which utilizes grammatical relations among words to semantically remove offensive content in a sentence. Specifically, for each sentence within a user message, we first identify offensive words, and then extract semantic relations and syntactic relations among words in the sentence. Based on the extracted relations, we estimate which words should be removed with those offensive words using two heuristic rules, i.e. "Modification Relation Rule" and

[1]Facebook statistics, at http://www.facebook.com/press/info.php?statistics

[2]The Internet Content Rating Association (ICRA), at http://www.icra.org/sitelabel/

CEAS 2010 - Seventh annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference July 13-14, 2010, Redmond, Washington, US

"Pattern Integrity Rule". To avoid confusion, we use the term "*removable*" to describe the result of our estimation. Words estimated as removable will be deleted from sentence at the end of filtering.

Compared with existing automatic filtering approaches, such as keyword censoring approach applied on YouTube website and content control approach applied in Microsoft Parental Controls, the proposed semantic filtering approach is able to remove offensive content in a text message thoroughly while keeping inoffensive content untouched as much as possible. Further, the readability of filtered content is guaranteed so as to make our filtering transparent to reader. Compared with manual filtering, which outputs optimal filtering results, the proposed semantic filtering approach is fully automatic with close filtering results.

To demonstrate the performance, we have created a dataset containing 11670 user comments collected from YouTube website. For each user comment, we perform both manual filtering and semantic filtering and then compare their outputs. According to experimental results, the semantic filtering achieves as high as 90.94% agreement with manual filtering. Meanwhile, the processing speed of semantic filtering is about 48.8 *msec* per comment, making it practical to be deployed at the server side of online websites. Furthermore, for user side application, we present an implementation of semantic filter as a Firefox extension, which is able to filter the user comments on webpages when a user is browsing on OSN websites. We show the advantage of our proposed filtering extension by comparing it with the "*Hide objectionable words*" function provided by YouTube.

2. RELATED WORK

In this section, we review related work and list some major approaches and methodologies for offensive language filtering in online communities. To demonstrate our observations, we present examples of applying different approaches in Table 1. Sentences in the table are cited from real user comments in YouTube dataset. For mental health of readers, we replaced the offensive words by special words.

2.1 Keyword Censoring Approach

Keyword censoring approaches match words appearing in text messages with offensive words stored in the blacklist. Once found, these offensive words will be removed, partially replaced (e.g., "a***"), completely replaced(e.g., "****"), or substituted by family friendly words (e.g., "nice"). Because of its simplicity, keyword based censoring approach has been widely applied in OSN websites, such as YouTube[3] and World of Warcraft[4].

However, the filtering result is not as desired. Brutally removing words from text message breaks the readability of text messages. Replacing offensive words with symbols usually makes it easy to guess the original offensive words. The idea of substitution seems tempting, but accurate substitution is usually impractical. Inaccurate substitution will introduce additional issues. For example, in 2001, Yahoo! deployed an Email filter which may automatically alter certain words in emails by family friendly words. This filter was criticized as a "foolish filter" by BBC news[5] because of its inaccurate substitution.

To demonstrate the shortcoming of keyword censoring approaches, we present examples in Table 1. According to presented filtering results, readers can still easily understand what the offender wants to say and even be able to infer the removed words. This indicates the failure filtering because offensive opinion has been successfully delivered to victims. Also, removing words from a sentence without considering their context breaks the readability of rest of the sentence.

Compared with keyword censoring approaches, our proposed semantic filtering approach is much more sophisticated and can achieve thorough filtering effort by utilizing the grammatical relations among words in the sentence. Given a sentence containing both offensive and inoffensive words, not only offensive words but also inoffensive words assisting to express offensive opinions will be removed during our filtering. In this way, we essentially stop the delivery of offensive opinion. And, there will be no way to infer the offensive content in original messages after filtering.

2.2 Content Control Approach

Content control approaches are usually deployed at user side or ISP side to prevent user from seeing inappropriate content on the Internet. Its filtering is usually done based on certain criteria, such as URL address, the occurrence of offensive words, and topic classification. Here our focus is text based criteria. For example, in Table 1, we present a sentence based content control approach with threshold set as the number of offensive words in the sentence. If at least two offensive words are detected within a sentence, the filter will remove the sentence from user message.

However, content control approaches are too coarse-grained to be applied in online communities. First of all, offender can easily bypass the filtering as long as knowing the estimation criteria. More important, a sentence in user comment may contain both offensive and inoffensive content. Inoffensive part may be removed falsely because of offensive part, e.g., the partial offensive case shown in Table 1. Not allowing user to post inoffensive content would easily drive users away and thus affect the growth of community.

Compared with content control approaches, we provide a fine-grained filtering by removing only the smallest syntactic part in the sentence containing offensive language. The inoffensive content in the original message will remain; thereby, user still has the freedom of speech for posting inoffensive content. We believe such delicate filtering will be more acceptable to online communities.

2.3 Manual Filtering Approach

Manual filtering is believed to produce the best filtering result. Basically, user messages are reviewed by community administrator before being posted on the website. As shown in Table 1, the administrator is able to easily understand what the author wants to express and precisely remove only the offensive content within the text.

However, manual filtering is very time and labor consuming, making it impossible to be widely applied. For example, in the Sina blog community[6], the blog administrator will manually review and filter user comments on some celebrities' public blogs. Obviously, users would expect a delay between posting a comment on a blog and displaying this

[3]YouTube, at http://www.youtube.com
[4]World of Warcraft, at www.worldofwarcraft.com/
[5]http://news.bbc.co.uk/2/hi/sci/tech/2138014.stm
[6]Sina Blog Community, at http://blog.sina.com.cn

Table 1: Filtering results with different approaches

	Sentence (we use *crying* and *pig* to denote two offensive words)		
	Partial Offensive		Absolute Offensive
Original Comment	"this video is *crying* good"	"it is aston martin and you are a *crying pig*"	"you're a *pig*"
Keyword Censoring	"this video is $c**$ good"	"it's aston martin and you are a $c*** p**$"	"you're a $p**$"
Content Control (thld=2)	"this video is *crying* good"	" "	"you're a *pig*"
Manual Filtering	"this video is good"	"it's aston martin"	" "

comment on the blog's webpage. Further, the filtering totally relies on the judgment of the community administrator.

Our proposed semantic filtering approach mimics the procedure of manual filtering by trying to understand the relations among words in order to remove the offensive content semantically. In our experiments, we show that the results between our proposed approach and manual filtering are very close (more than 90% agreement). Moreover, the proposed semantic filtering approach is fully automatic requiring no interference of administrator.

3. PROPOSED FILTERING PHILOSOPHY

The goal of our semantic filtering is to achieve filtering results close to that of manual filtering. To reach this goal, the foremost thing is to answer the question about how the filtering should be performed in order to get the desired filtering results.

In this section, we present our answer in three steps. First, we analyze the characteristics of offensive text content in user messages. Then, we introduce our filtering philosophy according to the summarized characteristics. Finally, we show how this philosophy is transformed into heuristic rules applicable in the filtering process.

3.1 Offensive Language Text Content

Based on the observation on user comments collected from YouTube website, a sentence in a user message may contain both offensive and inoffensive text content. Offensive text content is exposed intentionally with purpose of bringing negative influence to victims (e.g., the readers of text message). The victim receives the negative influence by reading the offensive part of sentence and understanding the carried offensive information.

Hence, the information carried by original sentence can be represented as $I = I_{off} + I_{inoff}$. The offender reaches his goal when the offensive information I_{off} is delivered to readers. Therefore, to achieve a thorough filtering, all words used to deliver I_{off} should be removed. Meanwhile, with respect to free speech, the part with I_{inoff} should be saved.

3.2 Filtering Philosophy

According to the analysis, we propose the philosophy that should be followed in sentence-level offensive language filtering:

- Precisely identify all offensive contents and remove them **semantically**, so that viewers will not notice the existence of offensive language in the original sentence;
- Keep the **readability** and inoffensive content in the sentence, so that the author will still be allowed to express his opinion freely as long as it is not offensive;

We call this the philosophy of "***filtering instead of blocking***". To the filter, the philosophy states that: if removing one word will make another word meaningless or confusing to readers, we should consider removing both words to keep the readability of a filtered sentence; meanwhile, we only remove words that are affected by offensive words.

For example, in the sentence "it is aston martin and you are a crying pig", suppose "crying" and "pig" are two offensive words, the sentence can be separated into two parts. The first part, "it is aston martin", is inoffensive; but the second part, "you are a crying pig" is offensive. Therefore, we should remove the second part completely while keeping the first part. The word "and" should also be removed in order to keep the transparency of filtering as well as the readability of filtered text content.

3.3 Filtering Rules

Specifically, the proposed philosophy is transformed into two heuristic rules to estimate the impact of removing words in a sentence.

Rule 1. (Modification Relation) in a modification relation, if the modifier is determined to be offensive, removing modifier solely is enough; if the head is determined to be offensive, both the head and the modifier should be removed.

The modification relation is a binary semantic relationship between two syntactic elements, such as word, phrase, etc. One element is named *head* and the other is named *modifier*. The modifier is used to describe the head (i.e. the modified component). Semantically, modifiers describe and provide more accurate definitional meaning for head. As the modifier acts as a complement, the removal of the modifier typically will not affect the grammaticality of the construction. For example, in the sentence "she likes red apples.", the adjective "red" is used to modify the noun "apples". Removing "red" will keep the readability of rest of sentence. We admit that, removing modifiers will lose some information carried by modifiers. However, if the modifier is determined removable but the head is not, removing modifier will remove only the offensive information.

Rule 2. (Pattern Integrity) if removing the offensive word breaks the integrity of sentence's basic pattern, the whole sentence should be removed in order to keep the readability.

English sentences and clauses are organized in basic patterns, such as "Subject-Verb", "Subject-Verb-Object", "Subject-Verb-Adjective", "Subject-Verb-Adverb", and "Subject-Verb-Noun". Every sentence or clause can be categorized into one pattern. The integrity of basic pattern is essential to the readability of content. For example, the sentence "she sleeps on the sofa." follows "Subject-Verb" pattern. If we

only remove "sleeps", the rest of the sentence, "she on the sofa." will become nonsense.

In the next section we will show details about applying these two rules during the filtering.

4. IDENTIFY REMOVABLE CONTENT BY GRAMMATICAL RELATIONS

A text message can be decomposed into a sequence of sentences. Each sentence is considered as a unit in filtering. Given a sentence containing both offensive words and inoffensive words, the goal of filtering is to identify inoffensive words which should be removed together with offensive words. We define the words that should be removed by the filtering as "*removable*" words.

We noticed that manual filtering can easily achieve this goal because human can easily understand the context of words in a sentence and precisely identify which words should be removed with known offensive words. So, we mimic the manual filtering in that, we extract the grammatical relations among words from sentences and use the proposed filtering rules to estimate the impact of removing offensive words on other inoffensive words based on extracted grammatical relations.

Specifically, the proposed approach includes two steps (details to be elaborated in the next two subsections). In the first step, we scan the sentence and see if offensive words exist. If exist, we continue to retrieve grammatical information (i.e. Part-of-Speech tags and typed dependency relations) among words in the sentence. Using retrieved grammatical information, we create a tree data structure, named *RelTree*, for the second step estimation. In this second step, we propose a set of estimation functions following the filtering rules we proposed. Using the RelTree structure and the proposed rules, we then estimate if there are inoffensive words that should be removed together with those identified offensive words.

The overview idea of our semantic filtering approach is shown in Algorithm 1. Within the algorithm, the functions $POStagging$ and $TDgenerator$ generate Part-of-Speech tags and typed dependency relations, respectively. We use existing NLP (Natural Language Processing) tools [2] to implement these two functions. In the rest of this section, we will focus on the design of two other functions $CreateRelTree$ and $EstimateRelTree$.

Note that, in this section, we assume that the filtering is based on a comprehensive offensive lexicon containing all offensive words. Words do not appear in the lexicon are considered inoffensive. We discuss the case of incomplete lexicon separately in Section 6.

4.1 First Step: Grammatical Analysis

In the first step, we extract two types of grammatical information from a given sentence. One is the Part-of-Speech information associated with every word. The other is the dependency relation among words. Part-of-Speech information helps us to understand the organization of a sentence, which is essential for keeping the readability when we try to remove words from a sentence. Dependency relations will be used directly to estimate the impact of removing one word on other semantically related words, making the filtering more "meaningful". Combining these two types of information, we can create a new data structure, called *RelTree*, for the next-step estimation.

```
   input  : a text comment T,
            a blacklist of offensive words Blacklist
   output: a filtered text comment T'
 1 T' ←"";
 2 senList ← chunk T into a list of sentences;
 3 foreach sentence s ∈ senList do
 4     scan s for offensive words using Blacklist;
 5     if no offensive word found then
 6         T' ← T' + s;
 7     end
 8     else
 9         PTree ← POStagging(s);/*get parse tree*/
10         TDset ← TDgenerator(s); /* get typed
           dependency relations */
11         RelTree ← CreateRelTree(PTree, TDset);
           /* create RelTree */
12         LabelRelTree ←
           EstimateRelTree(RelTree, Blacklist); /*
           estimate using RelTree */
13         s' ← remove all words in LabelRelTree those
           are labeled as "removable";
14         T' ← T' + s';
15     end
16 end
17 Return T';
```

Algorithm 1: Procedure of Semantic Filtering

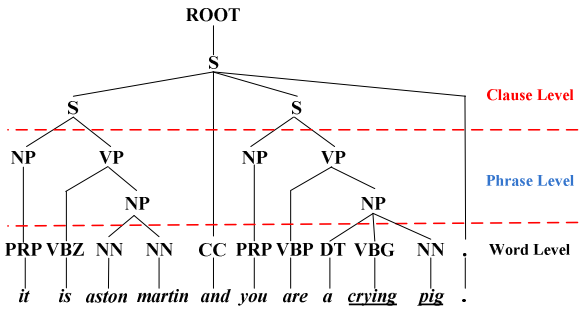

Figure 1: A parse tree of a sentence basing on Part-of-Speech tags

4.1.1 Part-of-Speech Tagging

Part-of-Speech tagging has been widely used in NLP applications to identify the syntactic properties of lexical items in a sentence, such as word or phrase. Through Part-of-Speech tagging, the sentence can be represented in a tree structure basing on Part-of-Speech tags. We adopt the Penn Treebank tag set [4] for our Part-of-Speech tagging.

An example of Penn Treebank style parse tree is shown in Figure 1. Here, the leaf nodes are words appearing in the sentence. The non-leaf nodes represent syntactic elements, such as phrases or clauses. Each element consists of the words within its subtree. For example,in Figure 1, the words "is", "aston", and "martin" constitute a Verb Phrase (i.e. VP) node. To avoid showing offensive language in this paper, we use two terms, "crying" and "pig", to replace original offensive words in this example.

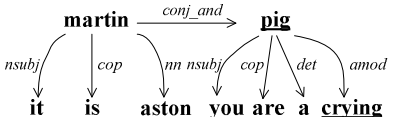

Figure 2: An example of typed dependency graph

4.1.2 Typed Dependency Relations

Typed Dependency is a kind of general relations describing the grammatical dependencies within a sentence, proposed by Stanford Natural Language Processing Group [6]. According to [6], each typed dependency includes a dependency type and a $(governor, dependent)$ word pair. For example, in the sentence "you are a crying pig.", the typed dependency *amod(pig, crying)* means that "crying" is an adjectival modifier of an noun phrase containing "pig". A typed dependency may represent the dependent relations between two syntactic elements, not limited to words only.

The typed dependencies in a sentence can be represented as a graph. For example, Figure 2 shows the typed dependency relations for the same sentence shown in Figure 1. We explain the relations appeared in Figure 2 from left to right: the nominal subject relation, *nsubj(martin, it)*, means that "it" is the syntactic subject of the clause (same is *nsubj(pig,you)*); the copula relation, *cop(martin, is)*, means that "martin" is the complement of verb "is" (same is *cop(pig, are)*); the noun compound modifier, *nn(martin, aston)*, means that the noun "aston" serves to modify the head noun "martin"; the determiner, *det(pig,a)*, means that "the" is a determiner of "pig"; the adjectival modifier, *amod(pig,crying)*, means that "crying" serves as adjectival modifier of "pig"; and the conjunct, *conj_and(martin, pig)*, means that the coordinating conjunction "and" is used to connect two elements with head "martin" and "pig", respectively.

4.1.3 Relation Tree (RelTree)

Both Part-of-Speech and typed dependency relations are utilized in the second step estimation. The parse tree shows the sentence syntactic organization and typed dependency relations provide semantic information among words. To combine both information, we propose a new data structure called *RelTree*.

In a RelTree, as shown in Figure 3, the leaf nodes are words in the sentence. And the non-leaf node represents either a phrase or a clause inside the sentence. In each non-leaf node, we associate the set of typed dependency relations on the words within its subtree. Each node only contains the typed dependency relations which have not appeared in its subtree nodes.

The RelTree data structure is proposed only for the convenience of offensiveness estimation in the next step. Algorithm 2 shows the algorithm for RelTree construction. With the parse tree $PTree$ given, the computational complexity of algorithm $CreateRelTree$ relies on the post-order traversal and the search in $TDset$. As the number of relations never exceeds $N(N-1)/2$, where N is the number of words in the sentence, the computational complexity is $O(N^3)$. The computational complexity itself is acceptable. Indeed, there are a lot of ways to improve the efficiency in the implementation of this algorithm.

4.2 Step Two: Bottom-up Estimation

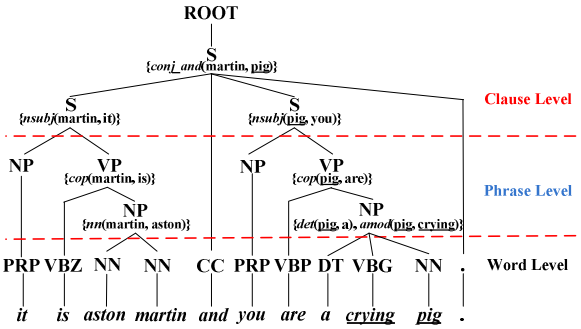

Figure 3: A RelTree combining the parse tree and typed dependency relations

input : a parse tree $PTree$,
a set of typed dependency relations $TDset$
output: a RelTree $RelTree$

```
1  RelTree ← PTree;
2  Remove all word nodes in RelTree;
3  Traverse RelTree in postorder  foreach node n
   visited do
4      if n is a leaf node then
5          n.wordset ← {n};/*create word nodes*/
6      end
7      if n is not a leaf node then
8          n.wordset ← ∅;
9          foreach direct child node c_i do
10             n.wordset ← n.wordset ∪ c_i.wordset;
11             n.rel ← ∅;
12             foreach relation T_i(G_i, D_i) in TDset do
13                 if G_i ∈ n.wordset and
                   D_i ∈ n.wordset then
14                     n.rel ← n.rel ∪ T_i(G_i, D_i);
15                     TDset ← TDset − T_i(G_i, D_i);
16                 end
17             end
18         end
19     end
20 end
21 Return RelTree;
```

Algorithm 2: create a RelTree using the parse tree and typed dependency relations

In the second step, we first use the offensive lexicon to identify offensive words in the sentence. The leaf node with an offensive word will be labeled as "removable". Starting from leaf nodes, we perform bottom-up estimation through a postorder traversal on the RelTree.

For each non-leaf node in the RelTree, we estimate whether it should be removed based on (1) the associated typed dependency relations and (2) its child nodes within its subtree. If a non-leaf node is estimated to be "removable", all its descendants, including words, within its subtree will also be labeled as "removable". The meaning of "removable" to a non-leaf node is that all words, phrases, or even clauses within its subtree have been determined to be removed at the end of filtering. The estimation process includes two steps. We first estimate based on typed dependency relations, and then apply a set of heuristic rules as complements.

4.2.1 Estimation with Typed Dependency Relations

Consider a non-leaf node n in a RelTree with a set $n.rel$ of typed dependency relations. Each relation describes a semantic connection between a *governor* word and a *dependent* word. Both words are leaf nodes in the subtree rooted at n. $n.rel$ could be empty when n only has one child node. For each typed dependency relation in $n.rel$, we study its semantic information and map it to an estimation function we defined. There are totally 55 typed dependency relations defined in [6], and we map each of them to one estimation function. We show all the mappings in Table 2.

These estimation functions and mapping are created following the Modification Relation and Pattern Integrity rules. Take the *Direct Object (dobj)* relation for example. In [6], the *dobj(G, D)* relation is defined as : the direct object of a verb phrase, containing governor word G, is the noun phrase, containing dependent word D. For example, in a relation $dobj(win, match)$, "win" is the governor word and "match" is the dependent word. According to Pattern Integrity rule, we know that "Subject-Verb-Object" is a basic pattern. Therefore, if either the phrase with G or phrase with D will be removed because of offensiveness, both phrases should be removed together.

To formalize, we define an estimation function *H(T) = H(P(G)) OR H(P(D))* and map relation $dobj(G, D)$ to it. We use symbol $C(G)$ and $P(G)$ to denote the clause and phrase containing word G as head, respectively. In this estimation function, $H(T)$ is the label to be assigned to relation T and $H(P(G))$ is the label with phrase node containing G in the RelTree. For example, in Figure 3,$P(aston)$ is the NP node of "aston martin", and $C(aston)$ is the clause (i.e. S) node of "it is aston martin.". The relation *dobj* will be labeled as "removable" if at least one of these two phrases are labeled as "removable".

Using the estimation function, we generate a label for every relation associated with node n and then for the node itself. If a relation $T(G, D)$ of node n is estimated and labeled as "removable", the two child nodes of n, containing word G and word D, will be labeled as "removable". If all relations in $n.rel$ are labeled as "removable", the node n as well as all its descendants, will be labeled as "removable".

4.2.2 Estimation with Heuristic Rules

Through experiments on YouTube dataset, we realize that filtering with typed dependency relations may not be enough. Heuristic rules must be applied as complement after typed dependency relation estimation. Applying heuristic rules is necessary mainly because of two reasons. First of all, the typed dependency relation contains some syntactic information but limited. For example, the possessive ending (i.e. *POS*) tag , which is a quite popular Part-of-Speech tag, is ignored during the typed dependency tagging.

Secondly, not all relations between syntactic elements in a sentence can be classified into one of typed dependency relations defined in [6]. For those uncertain relations, [6] defines a generic grammatical relation, named *dep*. To prevent confusion to filter, in the Table 2, we include *dep* into the Rule *H(T) = H(G) AND H(D)* which means either G or D is labeled removable will not affect each other and the label of T. Because *dep* relation stands for uncertain relation, we have to rely on Part-of-Speech tags in the RelTree for our filtering.

We present several examples of heuristic rules applied in

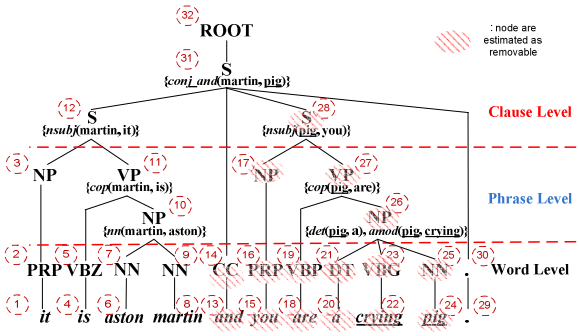

Figure 4: Estimate a RelTree in a bottom-up manner

semantic filtering in Table 3. Take the *conj* tag node rule as an example. The conjunct relation (*conj*) is a type of relation between two syntactic elements connected by a coordinating conjunction, such as "and". The parameters of *conj* do not include the coordinating conjunction. However, explicitly, the coordinating conjunction sits between the two parameters of *conj*. If one side is determined removable, the coordinating conjunction should be removed as well. For example, in the sentence "I like A and B", if either A or B is removed, the coordinating conjunction "*and*" should also be removed.

4.2.3 Estimation Algorithm

To estimate and assign labels for all nodes in a RelTree, we perform the estimation also in a bottom-up manner. Figure 4 shows an example estimation process. The number in the circle represents the order of estimation for each node in the RelTree. The dashed nodes are estimated as "removable". For example, the clause node with *nsubj(pig, you)* is estimated as "removable" according to the estimation. Therefore, its two child nodes containing "pig" and "you" respectively are both labeled as "removable". Moreover, the word "and" is removable according to the heuristic rule (i.e. *conj tag node rule*), in order to keep the filtering transparent to readers. Finally, inoffensive words, "you", "are", and "a", are removed with two offensive words, "crying" and "pig", in the filtering.

According to Algorithm 2, each typed dependency relation will appear exactly once in the RelTree. No relation will be checked repeatedly in the estimation. The cleaned sentence after filtering in this example will be "it is aston martin.". As we can see, the result satisfies the requirement of our proposed filtering philosophy. Only the offensive part, "you are a crying pig", is removed. The reader can still get the inoffensive information. The detailed algorithm for estimation process is presented in Algorithm 3.

5. APPLICATIONS

The proposed semantic filtering approach can be applied in real-world applications. In this section, we present two types of applications for demonstration. One application is at administrator side where the semantic filtering approach can help administrators to automatically remove offensive language in text messages submitted by users. The other application is at browser side where we implement the semantic filter as a Firefox extension for parental control.

Table 2: The mapping between typed dependency relations and estimation functions. (please see Section 4.2.1 for the definition of notations)

Index	Estimation Function	Typed Dependency Relation
1	H(T) = H(P(G))	cop, expl, measure, partmod, poss, possessive, preconj , prep/prepc, purpcl , quantmod, rcmod, ref , tmod;
2	H(T) = H(P(D))	pcomp, pobj , predet;
3	H(T) = H(C(G))	complm, mark, rel;
4	H(T) = H(P(G)) OR H(C(D))	xcomp;
5	H(T) = H(C(G)) OR H(P(D))	xsubj;
6	H(T) = H(G) OR H(P(D))	nsubj, nsubjpass;
7	H(T) = H(G) AND H(D)	conj, nn, number, dep;
8	H(T) = H(G)	aomp, advcl, advmod, agent, amod, appos, attr, aux, auxpass, cc, ccomp, det, neg, num, parataxis, punct;
9	H(T) = H(G) OR H(C(D))	csubj, csubjpass;
10	H(T) = H(P(G)) OR H(P(D))	abbrev, dobj, infmod, iobj, prt;

Table 3: Examples of heuristic rules applied in the proposed semantic filtering

Index	Name	Rules
1	POS tag node rule	**IF** {its previous noun is removable} **THEN** {POS tag node will be removable}
2	conj tag node rule	**IF** {both neighbor nodes are removable} **THEN** {conjunction node will be removable}
3	ADVP-VP rule	**IF** {the current non-leaf node has only two child nodes; one is ADVP node and the other is VP node} **THEN** { **IF** {ADVP node is removable} **THEN** {VP node will not be affected}; **IF** {VP node is removable} **THEN** {ADVP node will also be removable}; }

input : a RelTree *RelTree*,
a blacklist of offensive words *Blacklist*,
output: a labeled RelTree *LebelRelTree*

```
1  LebelRelTree ← RelTree;
2  Label all leaf nodes with offensive words by
   "removable" in LabelRelTree ;
3  Traverse LabelRelTree in postorder  foreach node
   n visited do
4      if n is a leaf node then
5          ignore; /* already labeled */
6      end
7      if n is not a leaf node then
8          if n only has one child node then
9              n.label ← n.child.label;
10         end
11         if n has more than one child node then
12             Estimate the label for n by its associated
               labels, using proposed estimation
               function and heuristic rules;
13         end
14     end
15 end
16 Return LabelRelTree;
```

Algorithm 3: estimate nodes in RelTree

5.1 Semantic Filter for Administrators in Online Communities

When a user wants to post a text message, he has to first send the message to the server and let the server post this message so that other users can see it. Therefore, one important place to eliminate offensive language in online communities is at the server of online communities. Many online communities have deployed sensor tools, such as swear filter in the World of Warcraft, to detect and remove sensitive words, including offensive words. However, as we shown in Section 2, filtering without considering semantics of text message turns out to be ineffective to fight against offensive language.

To a server side filtering tool, three metrics are most important to measure its performance: *effectiveness*, *accuracy*, and *speed*. For effectiveness, we already show the advantage of applying semantic filtering in Section 2. For accuracy and speed, we have implemented an offensive language filter in Java using the proposed semantic filtering approach. Stanford parser [2] is adapted to perform the Part-of-Speech tagging and generate the typed dependency relations.

In the following subsections, we first present the details about our collected YouTube dataset, and then present the results of our experiments.

5.1.1 YouTube Dataset

YouTube is a leading video sharing website in the world. Videos on YouTube websites are classified into 15 categories, such as *Comedy, Entertainment*, and *Music*. For each video webpage, contents contributed by users include the video for viewing, video title, video author ID, video description, and a list of text comments. Each text comment is associated with a user ID and a piece of text content. User ID identifies the author who generates this comment, and text content contains the body of user's opinion. Our filtering focuses on the text content of text comments.

To build the dataset, we collected text comments from video webpages on YouTube website in the days between September 27 and September 29, 2009. For each of 15 categories, we collected the top 20 "most discussed" video webpages ranked in "this week". For each webpage, we collected the first 40 text comments. The dataset contains 11670 text comments in total. For each text comment, we did manual filtering for each sentence in the comment. Specifically, the manual filtering process includes three steps. In

the first step, we read the comment and chunk it into sentences. Then, we identified the offensive words appearing in the sentence. Finally, for each offensive word, we marked the least set of words that should be removed together with it in order to eliminate the offensive information. We assign the collected YouTube dataset to five students for separate manual filtering. According to the results, 1739 text comments contain offensive words. Within these comments, 2063 sentences contain offensive words and the total number of unique offensive words appeared is 368.

During the manual filtering, one key question is how to identify offensive words. The same as in [3], the knowledge about offensive language in our case is represented by a lexicon of offensive words. All words in this lexicon have been determined to be offensive and should be prevented from being seen by readers. To estimate a word, our judgment is based on a list of offensive words provided by [1] and the word's meaning listed in the Urban Dictionary website[7]. All offensive words we detected are determined as offensive words by these two.

5.1.2 Accuracy

The results of both semantic filtering and manual filtering on a sentence will be in one of three types.

- "Clean": if there is no offensive word in the sentence;
- "Semantic Removing": there exist offensive words in the sentence and some inoffensive words have to be removed together with those offensive words;
- "Keyword Removing only": there exist offensive words in the sentence and removing those offensive words would be enough for the filtering.

To compare and measure the accuracy, we apply our proposed semantic filtering approach to process all comments collected in the YouTube dataset, and compare the result with the manual filtering result on the dataset. We assume that the manual filtering result represents the optimal filtering result we want to achieve. To understand the difference between the filtering result by human and by our filter, we manually compare the results of both filtering, sentence by sentence. The comparison outputs three types of results:

- if words removed by semantic filtering are exactly the same as those chosen by manual filtering, we call it "*Correct Filtering*";
- if more words are removed by semantic filtering than manual filtering, we call it "*Excessive Filtering*";
- if fewer words are removed by semantic filtering than manual filtering, we call it "*Insufficient Filtering*";

Excessive filtering means that there are unnecessary words, carrying inoffensive information, being falsely removed. Insufficient filtering means that filtering is not thorough.

According to our comparison, with 2063 sentences containing offensive words, the number of insufficient filtering is 58 (i.e. 2.81%), and the number of excessive filtering is 129 (i.e. 6.25%). To sum up, the overall ratios of accuracy of our implemented semantic filter is 90.94%.

Through reviewing the results, three reasons are responsible for the inaccuracy, which are *Informal English Writing*, *Incorrect Part-of-Speech tagging*, and *Incorrect Typed Dependency Analysis*.

[7]Urban Dictionary, at http://www.urbandictionary.com/

Informal English Writing is a character of language in the online community. People would like to write text comments on a casual or conversational tone, even with a lot of spelling errors. Luckily, our goal is not to understand the meaning of sentences but the relations among words. In the experiments, we apply the unlexicalized PCFG parser [7] proposed by Stanford NLP group and our results show some resistance to certain informal writings. For example, for word misuse case, in the Part-of-Speech tags generated from sentence "I like to apple you.", "apple" will be correctly tagged with as a verb. Another example, for misspelling case, the sentence of "I like to eat apple." and "I like to ea app." outputs the same Part-of-Speech tagging during our experiments. This is an important reason why we can still achieve such a high overall accuracy mentioned above.

However, piling up several sentences casually without using any delimiters, such as ".", "!", "?", will make it difficult for the parser to generate correct Part-of-Speech tags and typed dependency relations. Moreover, misspelling of certain sensitive words will confuse the parser (even human). For example, in some text comment, "your" is used to stand for "you're". There is no way for us to tell that.

Incorrect Part-of-Speech tagging contributes to 69.5% of incorrect filtering results. The major reason of incorrect tagging is caused by false sentence segmentation. In our experiments, we use the default sentence splitter provided by Stanford parser. Because typed dependency relations are generated based on the result of Part-of-Speech tagging, the incorrect tagging will certainly lead to incorrect filtering result. For example, many incorrect tagging cases are caused by arbitrarily inserting phrases in the pattern of "you *ABCD*" in a sentence. "*ABCD*" usually is a noun phrase or adjective phrase. Because "you" is a sensitive words in tagging, this can easily confuse the Stanford Parser.

Typed dependency analysis could be incorrect, even based on correct Part-of-Speech tagging. Especially, when the relation is uncertain, Stanford parser tends to assign a "dep" relation which basically means every thing could be possible. In some cases, we can apply heuristic rules mentioned in Section 4.2.2 as complement.

5.1.3 Speed

The processing speed on masses of text messages is important for filtering offensive language in real online communities. For our proposed filter, the time consuming part of filtering is the Part-of-Speech tagging and typed dependency relation generation parts. To measure the time efficiency of the implemented filter, we measure the speed in two cases: one is a normal case and the other is a high-overload case. In the normal case, we use the filter to process all text comments in the collected YouTube dataset. These comments contain both offensive comments and inoffensive comments. In the overload case, we apply the filter on offensive comments only. As the dataset is stored in files on local disk and each webpage is stored in a separate file, the measured processing time includes that for file I/O operations.

The time cost in each case for processing the same number of comments is listed in Figure 5. In both cases, the time cost increases almost linearly. In the high-overload case, the time cost is about 231.3 *msec* per comment. In the normal case, the time cost increases much slower because a lot of comments processed are inoffensive comments. For those inoffensive comments, no grammatical analysis is needed.

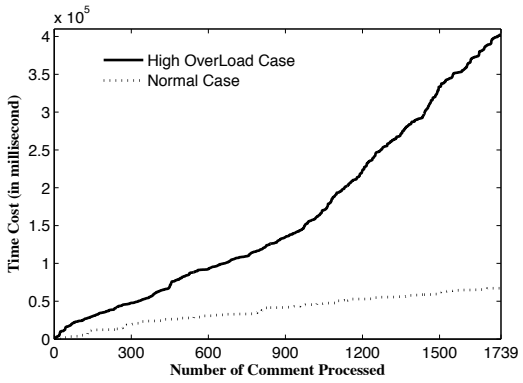

Figure 5: Measurement of Filtering Time Cost in Two Cases

Therefore, the average speed is much faster, about 38.6 *msec* per comment.

5.2 Firefox Extension for Parental Control

We also implemented the semantic filter as a Firefox extension which can be embedded into Firefox browser to filter offensive language in OSN websites. Briefly, our extension consists of three modules, which are *Pre-processor*, *Text Analyzer*, and *Post-processor*. The Pre-processor module is designed for content extraction. After receiving original webpage (e.g., a HTML file) from a website, it decomposes the content of a webpage and extract its text content posted by community members. The Text Analyzer module applies the proposed semantic filtering approach to filter the offensive language in the text content (Here we are interested in user messages). And the Post-processor module modifies the original webpage based on the results of filtering before showing the webpage in the browser.

The biggest challenge to implement the extension is to understand the structure of a webpage, because we are only interested in text contents that are contributed by members of online communities. Admittedly, to segment and label the text content inside HTML elements of a random webpage on Internet is still an open problem. Luckily, OSN websites usually have neat and fixed template for their webpages, such as YouTube, MySpace, and Facebook. All webpages in an OSN website share a limited number of templates. Users have no control over the structure of webpages. For example, if a user wants to post a comment on his friend's blog, he has to first send the text to the website and the website will update his friend's blog with submitted texts.

Based on this observation, we employ the Document Object Model (DOM) to extract the text content we are interested in. Each HTML webpage maps to a DOM tree where tags are internal nodes and the actual text, images or hyperlinks are the leaf nodes. For example, Figure 6 shows a comment with its corresponding fragments in the DOM tree. As we can see from Figure 6, in all webpages on YouTube website, comment body is within the subtree with the "watch-comment-body" tag. Situation is the same in other OSN websites. So far, we have considered extracting templates from websites, such as YouTube, Google Search, MySpace blog, Facebook, and Twitter. For example, every user status update on Twitter's *post wall* is within a node tagged by "entry-content".

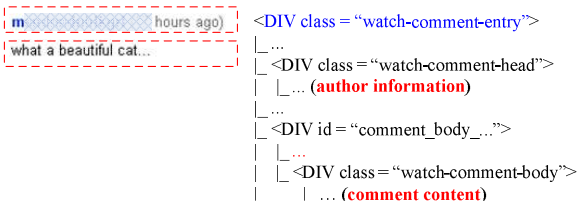

Figure 6: A YouTube comment and its corresponding DOM tree fragment

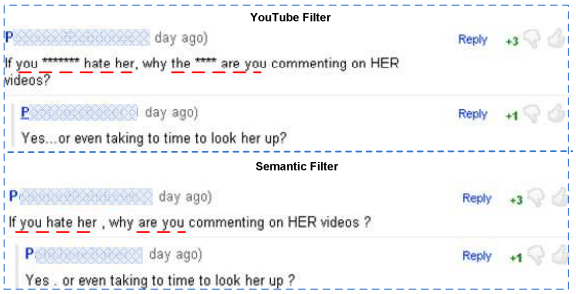

Figure 7: Comparison between YouTube filter and our semantic filter with partial offensive sentences

To demonstrate the effectiveness of semantic filtering with Firefox extension, we show a comparison of filtered results between our Firefox extension and YouTube's default filter (i.e. the "Hide objectionable words" function), in Figures 7 and 8. For better viewing, we put red lines at places with differences. As we can see from this comparison, the reader can easily guess the word filtered by YouTube's default filter. With proposed semantic filtering approach, we provide a "transparent" filtering. In these two cases, we are able to thoroughly remove the offensive parts while keeping the inoffensive part as much as possible.

We have also evaluated the speed of filtering ordinary video webpages on YouTube websites. During the evaluation, we select top 50 most "popular" videos in "This Week" and records the time cost for processing each webpage. Each webpage contains 10 user comments. According to the experiments, the minimum time cost for processing one webpage is 43 *msec* and the maximum is 2839 *msec*. The difference of time cost depends on the amount of text as well as the offensive words in the webpage. The mean of processing time is about 752 *msec*, 50% webpages requires less than 431 *msec* filtering time, and 75% webpages requires

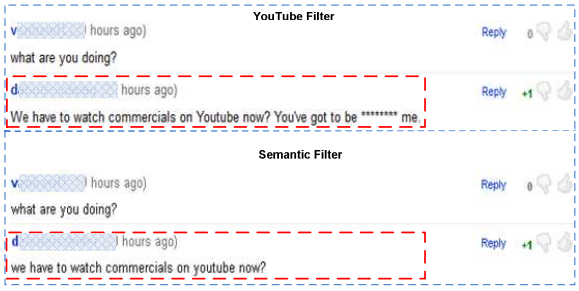

Figure 8: Comparison between YouTube filter and our semantic filter with absolute offensive sentences

less than 1211 *msec*.

6. LIMITATIONS AND COUNTERMEASURES

We now discuss (1) several limitations associated with our semantic filtering approach as well as a few possible ways offender may evade it, and (2) possible countermeasures against those evasions.

6.1 Offensive Language Detection

In this paper, we made an assumption that all offensive opinions are expressed by offensive words and we have a comprehensive offensive lexicon containing all offensive words. Based on this assumption, we adopt a simple word matching approach to identify offensive words in the sentence to be filtered. This assumption is made because *the focus of our paper is about offensive language filtering instead of detection*. Since our filtering approach depends on detection of offensive language, the filtering might fail if offensive language cannot be detected before the filtering process. To avoid filtering, offender may try to evade the offensive language detection mechanisms. For detection, there are many literatures discussing about detecting offensive language in sentence level [3] or message level [10]. For offensive lexicon generation, [9] presents a study. We believe offensive language detection a very challenging problem worthy of separate treatment.

6.2 Nature Language Process

The language used in online communities has its own characteristics, compared to the language used in Journal and newspaper. When applying NLP techniques, people would worry about the accuracy because of such characteristics as casual and informal English writing style. As we mentioned in Section 5.1.2, we did notice the inaccuracy brought by inaccurate text analysis. On the other hand, we discovered that language in online communities also has many characters suitable for text analysis. First of all, compared with articles or journals, most text messages posted are usually very short. A typical example is the $micro - blogging$ service provided by Twitter[8] which allows users to send brief text updates (less than 140 characters).

Secondly, most text messages use spoken English which has very simple and neat grammatical structures, making it easy to achieve high accuracy in text analysis. This phenomenon is also called *self-identity* in social psychology. It states that people actually like to make comments as simple and clear as possible so that readers can understand his/her opinion easily. In our case, *self-identity* makes the offensive comment easier to filter. If the comment is hard to analyze by a parser, it will probably cause difficulty to its human readers as well.

6.3 Changes to User Message

One concern about the proposed semantic filtering is the change made to the original text message. Admittedly, removing words from original message may cause information loss. Consider the information carried by original user message as $I = I_{inoff} + I_{off}$. During the filtering, offensive information I_{off} is okay to be removed, but the inoffensive information I_{inoff} should not be deleted or altered.

[8]Twitter, at http://twitter.com

In semantic filtering, we are fully aware of the impact of filtering. As stated in the proposed "filtering instead of blocking" philosophy, the semantic filtering only removes the smallest semantic part which containing offensive words in the sentence. Meanwhile, the readability of filtered sentence will be kept, making filtering transparent to user. Taking manual filtering as the standard, we demonstrate the ability to achieving close results by comparison in experiments.

7. CONCLUSION

Offensive language is a serious problem facing the online community. In this paper, we proposed a semantic filtering technique based on the grammatical relations of words in a sentence so that the rest of the filtered sentence is readable and the existence of offensive words in the original sentence is hard to notice. We tested the effectiveness of our approach with a large dataset and the results show that our techniques are very effective and accurate with little process overhead.

Our future work includes looking at the issues described in the discussion section. Moreover, as the most time-consuming part of semantic filtering is the sentence parsing process, we will examine other light-weighted NLP techniques to speed up sentence parsing. Last but not the lease, we also plan to extend our filtering approach to support other languages such as Chinese and French.

8. REFERENCES

[1] Bad word list and swear filter. *Available: http://www.noswearing.com/list.php*.

[2] The stanford parser: A statistical parser. *Available:http://nlp.stanford.edu/software/lex-parser.shtml*.

[3] M. K. Altaf Mahmud, Kazi Zubair Ahmed. Detecting flames and insults in text. In *Proceedings of 6th International Conference on Natural Language Processing*, 2008.

[4] A. Bies, M. Ferguson, K. Katz, R. Macintyre, M. Contributors, V. Tredinnick, G. Kim, M. A. Marcinkiewicz, and B. Schasberger. *Bracketing Guidelines for Treebank II Style Penn Treebank Project*, 1995.

[5] J. Cheng. Report: 80 percent of blogs contain "offensive" content. *ars technica*, 2007.

[6] M.-C. de Marneffe and C. D. Manning. *Stanford typed dependencies manual*, 2008.

[7] D. Klein and C. D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, 2003.

[8] H. Rheingold. *The virtual community : Homesteading on the electronic frontier*. Reading, Massachusetts: Addison-Wesley Publishing Company, 1993.

[9] J. Sjbergh and K. Araki. A multi-lingual dictionary of dirty words. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, 2008.

[10] E. Spertus. Smokey: Automatic recognition of hostile messages. In *Proceedings of the 9th Conference on Innovative Application of AI*, 1997.