# Improving Automated Controversy Detection on the Web

Myungha Jang and James Allan
Center for Intelligent Information Retrieval
College of Information and Computer Sciences
University of Massachusetts Amherst
{mhjang, allan}@cs.umass.edu

## ABSTRACT

Automatically detecting controversy on the Web is a useful capability for a search engine to help users review web content with a more balanced and critical view. The current state-of-the art approach is to find K-Nearest- Neighbors in Wikipedia to the document query, and to aggregate their controversy scores that are automatically computed from the Wikipedia edit-history features.

In this paper, we discover two major weakness in the prior work and propose modifications. First, the generated single query from document to find KNN Wikipages easily becomes ambiguous. Thus, we propose to generate multiple queries from smaller but more topically coherent paragraph of the document. Second, the automatically computed controversy scores of Wikipedia articles that depend on "edit war" features have a drawback that without an edit history, there can be no edit wars. To infer more reliable controversy scores for articles with little edit history, we smooth the original score from the scores of the neighbors with more established edit history. We show that the modified framework is improved by up to 5% for binary controversy classification in a publicly available dataset.

## 1. INTRODUCTION

The Web is an excellent source for obtaining accurate and useful information for a huge number of topics, but it is also an excellent source for obtaining misguided, untrustworthy and biased information. To help users review webpage contents with a more balanced and critical view, alerting users that the topic of a webpage is controversial will be a useful feature for a search engine.

Dori-Hacohen and Allan [4] proposed a framework for making binary classification on general webpage, whether the webpage presents a perspective on a controversial topic or not. Their framework consists of four steps:

1. **Matching k-NN Wikipages:** When a webpage is given as an input, they find $k$ nearest-neighbor Wikipages by generating a query from the 10 most frequent terms in the document.
2. **Computing Controversy Score on Wikipages:** From each of the $k$ Wikipage, they automatically extracted three controversy scores: C score [2], M score [10], and D score [4].
3. **Aggregate:** They aggregated the three types of $k$ scores using average or max operators.
4. **Vote and Classify:** They apply a voting scheme to turn the aggregated scores into a final binary decision, controversial or non-controversial.

While examining the performance of the current framework, we identified two major weaknesses. First, generating a single query from a document in Step 1 has issues. As documents almost always contain multiple sub-topics, the generated query contains an unknown mixture of different sub-topics. This makes the query's intent less clear, as it targets many sub-topics at the same time and in unknown balance. It is also unlikely that all sub-topics are covered in the query – or covered appropriately – because keywords are extracted from a bag-of-words, which does not model the existence of sub-topics as it is. As an alternative, we propose a text-segmentation based query generation approach named TILEQUERY. We first segment the document into multiple tiles, where each tile contains fewer sub-topics than the document, ideally one sub-topic per tile. We generate a query from each tile and then aggregate the ranked lists obtained from the tiles' queries. This can be viewed as a "divide-and-conquer" approach for document query generation.

Another issue is that the Wikipedia controversy scores depend on "edit-war" features, evidence of multiple editors exchanging opposing opinions. However, the controversy level is naturally underestimated on specific and sub-topical Wikipages whose topical disputes have often been delegated to other Wikipages of the broader topic. In other words, not having the "edit-war" does not necessarily mean that there was no war in this topic, but that the war has been happening somewhere else instead. This phenomenon causes the algorithm to easily make false negative errors (i.e., classifying "controversial" as "non-controversial").

We next provide details of the modified query generation approach and how we address the problem of missing or underestimated controversy scores using smoothing from neighbors. We then carry out an evaluation using 303 non-Wikipages as starting points and show the impact of our modifications to the framework on classification accuracy. We show that gains come from both changes but that correcting controversy scores has a greater impact, yielding 5% improvements in classification accuracy over the state-of-the art performance.



*SIGIR '16, July 17 - 21, 2016, Pisa, Italy*



ISBN 978-1-4503-4069-4/16/07…$15.00

DOI: http://dx.doi.org/10.1145/2911451.2914764

## 2. RELATED WORK

To the best of our knowledge, Dori-Hacohen and Allan were the first team to extend controversy detection to general open-domain web-pages [4]. However, many efforts have been made to understand and estimate controversy in Wikipedia. As Wikipedia contains manually tagged controversial articles by editors, machine-learning based methods approaches were trained to learn them. To estimate the level of controversy in Wikipages, information extracted from the edit-history, such as revision count, number of unique editors, number of reverts, the number of editors participating in the edit-war, and their reputations have been exploited [6, 10]. Sepehri Rad and Barbosa surveyed five established controversy detection algorithms on Wikipedia and compared their performances [8].

Outside of Wikipedia, controversy detection has also been studied within Twitter [7] or news corpus [1], focusing on political domain. Wang and Cardie recently studied online dispute detection using sentiment-analysis based method trained from controversial corpus in Wikiepdia [9]. However, as in Wikipedia, this method requires evidence of explicit disputes between two people, which is not applicable to general webpages.

## 3. MODIFIED FRAMEWORK

Here we provide details of the two modifications that we propose to the existing framework.

### 3.1 TileQuery Generation

***Document Segmentation*** We first use the block comparison algorithm described by the TextTiling technique [5]. The block comparison method defines as block with a few sentences, and computes a lexical similarity score for every gap between two blocks. When the similarity score dramatically changes at a gap, we assume that is where a sub-topic shift occurs and create a tile of blocks.

***Query Generation*** Once we create tiles from a document using TextTiling, we generate a query from each tile. There are often some tiles that are hard to understand its meaning without the context of the full text. Therefore, adding the global context helps clarify the topic of each tile, anchoring the tile's query to the containing document's topic. We hence generate a query by using the $g$ (global) most frequent terms from the document,and the $l$ (local) most frequent terms from the tile. We empirically found that $g = 3, l = 7$ gives the best performance when using 10 terms.

***Aggregating the Ranked Lists*** Each TILEQUERY returns a ranked list. We compute the relevance score for each retrieved Wikipage $w_i$ by aggregating the reversed ranking order:

$$Relevance(w_i) = \sum_{t_i \in T} (k - rank_{t_i}(w_i)) \qquad (1)$$

where $T$ is the set of tiles generated from the document. This scoring prioritizes Wikipages that appear high in some tile or at reasonable ranks in many tiles, or preferably both.

### 3.2 Smoothing Controversy Score of Wikipages

***Wikipedia Controversy Scores*** Previous work studied algorithms for automatically computing scores that estimate the level of controversy. They use features available in Wikipages, meta-data, talk pages, and edit-history. We briefly explain the three scores that the previous framework adopted.

**C Score** This is a regression-based method that estimates the revision count of Wikipages with {controversial} tags. The features include information from the edit-history, such as number of unique editors and number of reverts, as well as some metadata of Wikipages [6].

**M Score** Yasseri et al. investigated edit-wars based on statistical features of edits [10]. This score is theoretically unbounded ranging from 0 to a few billions.

**D Score** This is a Boolean value indicating whether a Wikipage contains a dispute tag in it, which is assigned by the page's contributors This dataset is extremely sparse, covering only 0.03% of the articles [4].

Unfortunately, these approaches are limited for the same reason that many Wikipages with controversial topics do not have sufficient edit-history or explicit edit-wars. There is a tendency that the heat of the edit-wars are focused on one Wikipage of a general and broad topic, leaving other related but sub-topical pages less attended. After all, there is simply no point of having the same "war" on all similar Wikipages. Table 1 shows an example of a few "abortion" related topics and their M and C Score. While the "Abortion" page received a lot of attention, other pages with more specific topics such as *Abortion in certain countries* and *Abortion Act* had virtually no edit-wars. Unless there is a specific issue or event specifically tied to the page, all general disputes on abortion have been delegated to the "Abortion" page.

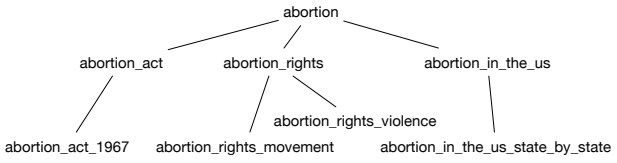


**Figure 1: An example of the constructed network for *Abortion***

Due to this phenomenon, even if we generate a better query to find more relevant $k$ pages, the framework still would not be able to fully take advantage of that due to the underestimated scores. Hence, it is necessary to revise these scores to reflect controversy better. If the purpose of the M or C score was to measure the controversy level presented in the Wikipage *per se*, we need newly revised scores that accurately signify controversiality of the topic in general. To do this, we will construct a network that links topically related neighbors within the Wikipedia. We then revise the controversy score by "smoothing" using the scores of neighbors with more edit history, whose scores were computed with more confidence.

***Constructing Wikipages' Graph*** We construct a tree-structured graph to identify topically related neighbors of a Wikipage. Let $G = (V, E)$ be a directed graph with nodes $V$ (Wikipages) and edges $E$ (sub-topical relations). An edge $e(u, v)$ represents that node $v$ is a sub-topic of $u$.

As a simple and straightforward method to construct the edges, we look at their titles. If a Wikipage $u$'s title is used as a prefix of other $v$'s title, we assume that $v$ is sub-topic of $u$. While we use nodes' titles to construct edges, we assume there is a mapping between a title and a node and will use them interchangeably.

Let a Wikititle T be a ordered list $[t_1, t_2, ..., t_n]$, where $t_i$ is an i-th space-delimited token. The parent node set P(T) (i.e., Wikipages whose titles that have $T$ as a child) is obtained by:

$$P(T) = \{P_T^i | P_T^i \in W_T, i \in \{1..n\}\}, P_T^i = concatenate[t_1, ..., t_i]$$

where $W_T$ is a set of all Wikipedia titles. The graph also contains many noisy relations when the prefix is an ambiguous entity, or a simply too general word, such as "American". To filter out the noisy relations, we remove the edges if two pages are not linked in any direction. Using this graph, we finally revise the controversy score using smoothing.

***Network-based Smoothing*** When Wikipage is given as a query, we extract a sub-graph around the node from the constructed graph using one of the two methods:

**Pair-based**: A sub-graph around the query node including its children at all depth and its parents. The resultant graph only consists of nodes that have a direct prefix-contain relation with the query node.

**Clique-based**: Pair-based sub-graph + sibling nodes that share the same parents with the query node. Although siblings may not be topically related to the query node especially if the parent (i.e., prefix) is a general term, this allows broader coverage of potentially related pages.

Once we obtain the sub-graph, we treat all nodes in the sub-graph as topically related neighbors of the query node. We want to fix the query node's controversy score by smoothing from neighbors that have more reliable scores. For that we assume that the controversy score of a Wikipage with more revision history is more reliable. Hence we convert this graph into a weighted, directed network whose direction represents which way influence should extend to (i.e., the one with higher revision count to the other with lower count), and whose edge weight represents the confidence of the influence relation, which is the revision count of the source (Figure 2). From the graph, the new controversy score of Wikipage $W_i$ is computed as:

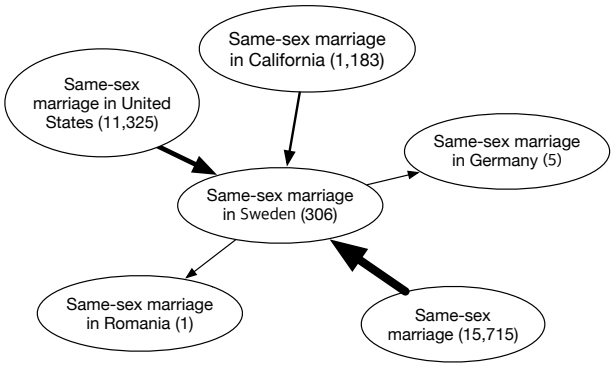


**Figure 2: A network constructed around the query "Same-sex marriage in Sweden". The scores are borrowed from three neighbors that have more revision counts than the query page by weighted smoothing.**

**Table 1: An example of two controversy scores on several Wikipages on "Abortion", before and after score smoothing**

| | Original scores | | Revised scores | |
|---|---|---|---|---|
| | M | C | M | C |
| Abortion | 4,102,593 | 0.300 | 4,102,593 | 0.300 |
| Abortion_Act | 0 | 0 | 0 | 0 |
| Abortion_in_China | 0 | 0 | 2,062,156 | 0.166 |
| Abortion_in_England | 0 | 0 | 2,128,909 | 0.172 |
| Abortion_in_the_US | 0 | 0.002 | 2,983,300 | 0.218 |

$$C'(W_i) = \sum_{W_j \in inLinks(W_i)} \frac{C(W_j) * r_j}{\sum r_k} \quad (2)$$

where $r_i$ is a revision count of $W_i$.

## 3.3 Aggregation and Voting

We summarize the aggregation and voting schemes introduced by previous work. Once the controversy scores are obtained for $k$ Wikipages, we aggregate the $k$ scores by taking average or max of them. Since we use three different scores, M, C, and D, three aggregated scores, $M_{agg}, C_{agg}$, and $D_{agg}$ are computed. We turn these scores into binary label indicating controversial (1) or non-controversial (0), using corresponding thresholds. $M_{label} = 1$ if $M_{agg} \geq Threshold_M$, and 0 otherwise. Using the three generated labels, we use a voting scheme to make a final decision. We test 6 voting schemes as parameters in our experiments.

The webpage is controversial if:

— **C/M/D**: $\{C_{label}, M_{label}, D_{label}\}$ is 1, respectively.

— **Majority**: the majority (i.e., at least two) of $\{C_{label}, M_{label}, D_{label}\}$ is 1.

— **Or/And**: $C_{label}\{\vee/\wedge\}M_{label}$ $\{\vee/\wedge\}D_{label}$ is 1.

# 4. EXPERIMENTS

## 4.1 Dataset

We use the publicly available controversy dataset released by Dori-Hacohen et al. [3]. We used 303 clueweb documents whose controversy level was annotated with four scales: 1 - "clearly controversial", 2 - "possibly controversial", 3 - "possibly non-controversial", and 4 - "clearly non-controversial". To convert the annotations to binary judgments, we treated the documents with average ratings among annotators of less than 2.5 as controversial, and otherwise non-controversial. Of 303 documents, 42% of them are controversial.

To test the effectiveness of the proposed query method, we consider two other query baselines. One is TF10, the 10 most frequent terms, as in the prior work. As taking only $k$ terms as in a query might miss information, we consider another baseline, ALL query that uses all terms in a document as a query to observe the extreme case of TF$N$.

We consider 9 settings from all possible combinations of three query methods and three scoring schemes (Table 2). Run 4 is the setting proposed in the prior work [4]. In each setting, we varied the four sets of parameters, the number of neighbors K (1, 5, 10, 15, 20), aggregation method (avg, max), voting methods (C, M, D, Majority, Or, And, D$\vee$(C $\wedge$ M)), and thresholds for C and M as tested in [4]. We found the best parameter setting for each run using 5-fold cross validation with the target metric accuracy. Thus, for 9

Table 2: Accuracy, F1, and the best parameters in 5-fold runs for different query and inferred score settings.

| Run | Query | Inferred Score | K | C Threshold | M Threshold | Aggregation | Acc. | F1 |
|---|---|---|---|---|---|---|---|---|
| 1 | | N/A | {5, 20} | $\{0.17, 4.18 \cdot 10^{-2}\}$ | {40000,20000} | {M, Maj.} | 0.72 | 0.50 |
| 2 | ALL | Clique | 15 | $\{0.17, 4.18 \cdot 10^{-2}\}$ | {40000,20000} | {M, Maj.} | 0.78 | 0.68 |
| 3 | | Pair | {5, 20} | $\{0.17, 4.18 \cdot 10^{-2}\}$ | {40000,20000} | {M, Maj.} | 0.73 | 0.53 |
| 4 | | N/A | 20 | $4.18 \cdot 10^{-2}$ | {20000, 40000, 84930} | {M, Maj.} | 0.75 | 0.57 |
| 5 | TF10 | Clique | 20 | $4.18 \cdot 10^{-2}$ | 84930 | Maj. | 0.79 | 0.68 |
| 6 | | Pair | {10, 20} | $4.18 \cdot 10^{-2}$ | {20000, 84930} | Maj. | 0.75 | 0.57 |
| 7 | | N/A | {10,15,20} | $4.18 \cdot 10^{-2}$ | {40000,20000} | {M, Maj.} | 0.75 | 0.59 |
| 8 | TILE | Clique | 20 | 0.17 | 40000 | M | 0.80 | 0.71 |
| 9 | | Pair | {10,15,20} | $4.18 \cdot 10^{-2}$ | {40000,20000} | {M, Maj.} | 0.75 | 0.61 |

Table 3: Improvements of accuracy and F1 Score between runs (bold: statistically significant)

| Row # | Runs | $\lvert Acc_1\text{-}Acc_2 \rvert$ | $\lvert F1_1\text{-}F1_2 \rvert$ | p value |
|---|---|---|---|---|
| 1 | 1 vs 2 | **6%** | **2%** | $0.01 \cdot 10^{-2}$ |
| 2 | 1 vs 4 | 3% | 7% | 0.61 |
| 3 | 1 vs 7 | **3%** | **9%** | 0.08 |
| 4 | 4 vs 5 | **4%** | **11%** | $0.17 \cdot 10^{-2}$ |
| 5 | 4 vs 7 | 0% | 2% | 0.18 |
| 6 | 5 vs 8 | **1%** | **3%** | $0.06 \cdot 10^{-2}$ |
| 7 | 6 vs 9 | **0%** | **4%** | $0.01 \cdot 10^{-2}$ |
| 8 | 7 vs 8 | **5%** | **12%** | $0.01 \cdot 10^{-2}$ |

settings, there are 5 sets of parameters learned for each fold. We used McNemar's Test[1] for statistical significance test.

## 4.2 Results

***Impact of Query Methods*** Our statistical significant tests suggest that the difference of accuracies between the three query methods in runs 1, 4, and 7 are not significant (Row 2 & 5 in Table 3), which suggest that the three methods mostly made similar classifications.

However, once we apply neighbor-based smoothing on controversy scores, query methods cause classification to work differently. The accuracy gain of TileQuery over TF10 was 1%, and 4% of F1-score with smoothing. Although the accuracy gain was small, the query set that each method performed well was different as the significance test implies (Row 6 & 7 in Table 3).

***Impact of Neighbor-based Smoothing*** In all settings, using controversy score smoothing significantly improved the classification accuracy and F1-score. As row 1, 4, and 8 in Table 3 show, the accuracy was improved by 4-6% and the F1-score was improved by 2-12% in all three query methods. Clique-based neighbor selection consistently outperformed pair-based selection.

# 5. CONCLUSION AND FUTURE WORK

We revisited the prior work for automatically detecting controversy from the general open-domain webpages. We identified two major weakness in the framework and proposed two modifications to fix the issues. The controversy score smoothing consistently improved the controversy classification accuracies by 4-6% compared to those without smoothing. Overall, the run with our two modifications of tilequery and controversy score smoothing gave the best accuracy improving the previous framework by 5%. In the future, we plan to further investigate different scenarios when TF10 and TileQuery works well. As we were only able to find topically related neighbors for 5% of the Wikipages with prefix-relation, we will explore more sophisticated methods to increase this coverage.

[1]https://en.wikipedia.org/wiki/McNemar%27s_test


# Acknowledgement

This work was supported in part by the Center for Intelligent Information Retrieval, in part by NSF grant #IIS-0910884, and in part by NSF grant #IIS-1217281. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor. The authors thank Shiri Dori-Hacohen for providing valuable resources.


# 6. REFERENCES


[1] Y. Choi, Y. Jung, and S.-H. Myaeng. Identifying controversial issues and their sub-topics in news articles. In *PAISI*, volume 6122 of *Lecture Notes in Computer Science*, pages 140–153. Springer, 2010.

[2] S. Das, A. Lavoie, and M. Magdon-Ismail. Manipulation among the arbiters of collective intelligence: how wikipedia administrators mold public opinion. CIKM '13, pages 1097–1106, New York, NY, USA, 2013. ACM.

[3] S. Dori-Hacohen and J. Allan. Detecting controversy on the web. In *CIKM*, pages 1845–1848. ACM, 2013.

[4] S. Dori-Hacohen and J. Allan. Automated controversy detection on the web. ECIR '15, pages 423–434, 2015.

[5] M. A. Hearst. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Comput. Linguist.*, 23(1):33–64, Mar. 1997.

[6] A. Kittur, B. Suh, B. A. Pendleton, and E. H. Chi. He says, she says: Conflict and coordination in wikipedia. CHI '07, pages 453–462, New York, NY, USA, 2007. ACM.

[7] A.-M. Popescu and M. Pennacchiotti. Detecting controversial events from twitter. CIKM '10, pages 1873–1876. ACM, 2010.

[8] H. S. Rad and D. Barbosa. Identifying controversial articles in wikipedia: A comparative study. WikiSym '12, pages 7:1–7:10. ACM, 2012.

[9] L. Wang and C. Cardie. A piece of my mind: A sentiment analysis approach for online dispute detection. ACL '14, pages 693–699, 2014.

[10] T. Yasseri, R. Sumi, A. Rung, A. Kornai, and J. Kertész. Dynamics of conflicts in wikipedia. *PLoS One*, 7(6):e38869, June 2012.