Detecting Wikipedia vandalism using WikiTrust

Bo Adler
Fujitsu Labs of America
and
UC Santa Cruz

Luca de Alfaro
Google Inc
and
UC Santa Cruz (on leave)

Ian Pye
CloudFlare Inc
and
UC Santa Cruz
Anyone can edit the Wikipedia

- This has been the key to its success (get knowledge from all sources).
- Unfortunately, this also leads to vandalism.
WikiTrust: A reputation system for wiki authors and content

- Authors gain reputation when their contributions are preserved by others.
- Text gains reputation when it is revised by multiple distinct high-reputation authors.
- WikiTrust computes the reputation of individual authors and words.
Revision quality

$r_j$ is good: $d(r_i, r_k) > d(r_j, r_k)$
"$r_j$ went towards the future $r_k$"

$r_j$ is bad: $d(r_i, r_k) < d(r_j, r_k)$
"$r_j$ went against the future $r_k$"
Revision quality measures the fraction of change that agrees with the future page evolution.

\[ q(r_j \mid r_i, r_k) = \frac{d(r_i, r_k) - d(r_j, r_k)}{d(r_i, r_j)} \]

- \( q(r_j \mid r_i, r_k) \approx +1 \): revision \( r_j \) was preserved by \( r_k \)
- \( q(r_j \mid r_i, r_k) \approx -1 \): revision \( r_j \) was reverted by \( r_k \)

**Corollary:** we can detect reversions automatically.
Author reputation

\[ q(r_j | r_i, r_k) = \frac{d(r_i, r_k) - d(r_j, r_k)}{d(r_i, r_j)} \]

Revision Quality:

The reputation update:

- increases if \( q(r_j | r_i, r_k) > 0 \).
- decreases if \( q(r_j | r_i, r_k) < 0 \).

The increase/decrease is greater, the greater the reputation of \( A_k \).
Author reputation predicts reversions

- **Recall:** Low-reputation authors (those in the bottom 20% of reputation) account for 18.1% of the edits, and for 82.9% of reverted edits.

- **Precision:** An edit has a 5.7% probability of being reverted. However, if the edit is done by a low-reputation author, this probability raises to 48.9%.
Text Reputation (a.k.a. text trust)

Compute trust at the individual word granularity.

- New text starts at reputation 0.
- When text of reputation $t$ is revised by an author of reputation $r > t$, the text can gain reputation $k(r - t)$.
- To prevent abuse, we mark every word of text with the last 3 authors who caused its reputation to rise. If an author appears in this list, se cannot rise the word reputation.
- Word reputation is displayed via text background color: the more intense orange, the lower the reputation.
Low word reputation predicts deletion

- **Recall wrt. deletions:** Text in the bottom half of reputation values constitutes 3.4% of the text, yet corresponds to 66% of the text that is deleted in the next revision.

- **Precision wrt. deletions:** Text in the bottom half of reputation values has a probability of 33% of being deleted in the very next revision, compared with 1.9% for general text. The probability raises to 62% for text in the bottom fifth of reputation values.

Data obtained by analyzing 1,000 articles selected at random among those with at least 200 revisions.
Word reputation predicts lifespan
Using WikiTrust for vandalism detection

Idea: since author and word reputation are both good predictors of revisions, can we build a vandalism-detection system on the basis of these, and a few other signals?

Challenge: we wanted to use ONLY signals that were easily available in the WikiTrust database. No additional NLP or other complicated analysis! Our question was: how well can we do with the signals we have readily available?
Two vandalism detection problems

- **Z: Zero-delay vandalism detection**: use only past data.
  - Use: is the edit just made vandalism?

- **H: Historical vandalism detection**: use data both in the past and future of the revision.
  - Use: given a page, what is a recent revision that is very likely not vandalism?
Features: reputation

- **Author reputation** \((Z, H)^*\)
- **Author is anonymous** \((Z, H)\)
- **Text reputation**: we compute the histogram of word reputation for a revision, and we consider:
  - The histogram of the word reputation \((Z, H)\).
  - The histogram of word reputation for the previous revision \((Z, H)\), normalized so all columns sum to 1.
  - The difference between the word reputation of the present, and of the previous, revision \((Z, H)\).

*: In the PAN 2010 Z evaluation, we did not use author reputation, since author reputation was available only for a later date than when the revisions were created.
Features: revision quality

- **Minimum revision quality** (H): the minimum value of edit quality, measured wrt. all past and future revisions considered.
- **Average revision quality** (H): the average value of edit quality, where $q(r_j | r_i, r_k)$ is weighed:
  - According to the reputation of the author of $r_k$
  - Checking that $d(r_i, r_j)$ is not too small compared with $\min[d(r_i, r_k), d(r_j, r_k)]$, otherwise the “judge” revision $r_k$ is too far from the judged revision, and the judgement is imprecise.
- **Delta**: extent of difference wrt. previous revision (dealing with block moves nicely).
Features: timing

- Time to the previous revision \((Z, H)\)
- Time to the following revision \((Z, H)\)
- Local time of day of revision (approximated as CST for logged-in users)

We also experimented with various other features, but these were not picked up by our classifier.
The classifier: ADT

We limited ourselves to the classifiers available as part of the Weka toolset.

We experimented with most of them, and the best was ADT. A small tree size sufficed: we saw no gains going from 10 to 20 boosting iterations.

Evidently, our performance was dominated by a few, very strong signals.

We used a weight-sensitive version of the classifier, where a coefficient $\beta$ was used to give more weight to the error of classifying vandalism as normal, rather than the other way round.
## Results

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Type</th>
<th>Nodes</th>
<th>$\beta$</th>
<th>Dataset</th>
<th>Recall</th>
<th>Precision</th>
<th>False Pos.</th>
<th>ROC area</th>
</tr>
</thead>
<tbody>
<tr>
<td>H10b20</td>
<td>Historical</td>
<td>10</td>
<td>20</td>
<td>Training</td>
<td>0.903</td>
<td>0.430</td>
<td>0.078</td>
<td>0.956</td>
</tr>
<tr>
<td>H10b20</td>
<td>Historical</td>
<td>10</td>
<td>20</td>
<td>Evaluation</td>
<td>0.835</td>
<td>0.485</td>
<td>0.082</td>
<td>0.934</td>
</tr>
<tr>
<td>H20b50</td>
<td>Historical</td>
<td>20</td>
<td>50</td>
<td>Training</td>
<td>0.950</td>
<td>0.276</td>
<td>0.163</td>
<td>0.957</td>
</tr>
<tr>
<td>H20b50</td>
<td>Historical</td>
<td>20</td>
<td>50</td>
<td>Evaluation</td>
<td>0.924</td>
<td>0.302</td>
<td>0.198</td>
<td>0.937</td>
</tr>
<tr>
<td>Z10b20</td>
<td>Zero-Delay</td>
<td>10</td>
<td>20</td>
<td>Training</td>
<td>0.883</td>
<td>0.286</td>
<td>0.144</td>
<td>0.930</td>
</tr>
<tr>
<td>Z10b20</td>
<td>Zero-Delay</td>
<td>10</td>
<td>20</td>
<td>Evaluation</td>
<td>0.828</td>
<td>0.308</td>
<td>0.173</td>
<td>0.909</td>
</tr>
<tr>
<td>Z20b10</td>
<td>Zero-Delay</td>
<td>20</td>
<td>10</td>
<td>Training</td>
<td>0.837</td>
<td>0.357</td>
<td>0.098</td>
<td>0.931</td>
</tr>
<tr>
<td>Z20b10</td>
<td>Zero-Delay</td>
<td>20</td>
<td>10</td>
<td>Evaluation</td>
<td>0.771</td>
<td>0.369</td>
<td>0.122</td>
<td>0.904</td>
</tr>
</tbody>
</table>

**Table 1.** Performance summary of the historical and zero-delay vandalism tools, evaluated on the training dataset (via 10-fold cross validation), and on the PAN 2010 evaluation dataset. The classifier used for the PAN 2010 submission is Z20b10.
Historical classification tree

: 0.134
  | (1)Min_quality < -0.662: 0.891
  |   | (3)L_delta_hist0 < 0.347: -0.974
  |   | (3)L_delta_hist0 >= 0.347: 0.151
  |   | (4)Max_dissent < 0.171: -1.329
  |   | (4)Max_dissent >= 0.171: 0.086
  |   |   | (10)Next_comment_len < 110.5: -0.288
  |   |   | (10)Next_comment_len >= 110.5: 0.169
  | (1)Min_quality >= -0.662: -1.203
  | (2)Reputation < 0.049: 0.358
  | (2)Reputation >= 0.049: -1.012
  |   | (6)P_prev_hist5 < 0.01: 0.482
  |   | (6)P_prev_hist5 >= 0.01: -0.376
  |   |   | (7)Avg_quality < 0.156: 0.5
  |   |   | (7)Avg_quality >= 0.156: -2.625
  |   |   | (9)L_delta_hist2 < 0.347: -0.757
  |   |   | (9)L_delta_hist2 >= 0.347: 1.193
  | (5)Logtime_next < 2.74: 1.188
  | (5)Logtime_next >= 2.74: 0.045
  |   | (8)Delta < 3.741: -0.255
  |   | (8)Delta >= 3.741: 0.168
Zero-delay classification tree

: 0.134
   (1) L_delta_hist0 < 0.347: -1.018
     | (7) Hist0 < 0.5: -0.113
     | (7) Hist0 >= 0.5: 0.528
   (1) L_delta_hist0 >= 0.347: 0.766
     | (3) L_delta_hist3 < 0.347: 0.026
       | (8) L_delta_hist4 < 0.347: 0.1
       | (8) L_delta_hist4 >= 0.347: -0.751
     | (3) L_delta_hist3 >= 0.347: -0.962
   (2) Anon = False: -0.576
   (2) Anon = True: 0.312
   (4) P_prev_hist9 < 0.115: -0.333
   (4) P_prev_hist9 >= 0.115: 0.182
     | (9) Hist7 < 1.5: 1.217
     | (9) Hist7 >= 1.5: -0.029
   (5) Delta < 2.901: -0.251
   (5) Delta >= 2.901: 0.182
   (10) Comment_len < 18.5: 0.123
   (10) Comment_len >= 18.5: -0.229
The WikiTrust vandalism API

- To obtain the probability of vandalism of revision 1234:

- To obtain all the signals we use to classify revision 1234:

- To select the best revisions for page 12:

WikiTrust: www.wikitrust.net