Tagging Stack Overflow Questions

A multi-class problem
The Problem

17m+ Questions on Stack Overflow

54k+ Tags on Stack Overflow
Challenges Deep-Dive

Number of tags
Over 54k tags
When the number of tags is increased, the odds of a proper selection are decreased.

Types of tags
Similar tag vocabulary
Many programming languages contain similar words or phrases

Filtering Text
Data in html format
The data was in an html format and included non-alphanumeric characters
In addition, challenge throughout to find words most related to tag
Solutions
Development Cycle
How to walk through solutions

1. Develop
2. Test
3. Refine
4. Analyze Results
The Algorithms

- Scoring
  - Tf(class) - Tf(total)
  - TF-IDF

- Filtering
  - POS
  - N-Grams
    - Adjectives
    - Adjectives
    - Verbs
    - Bi-Grams
## Naive Bias

Baseline for the rest of testing

<table>
<thead>
<tr>
<th>21%</th>
<th>Naive Bias</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Naive Bias to get a baseline for future test</td>
<td></td>
</tr>
</tbody>
</table>

- Filler words often determine guessed tag

<table>
<thead>
<tr>
<th>54%</th>
<th>Naive Bias with stop words filtered</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtering stop words flushing out important words</td>
<td></td>
</tr>
</tbody>
</table>

- Increased accuracy
- Higher look at important words in tagging
POS Tagging

Filtering Method

74% Nouns
Filtering out text so that only nouns are analyzed

Nouns are good indication of overall subject

47% Adjectives
Filtering out text so that only adjectives are analyzed

Adjectives across different programming languages can be very similar

55% Verbs
Filtering out text so that only verbs are analyzed

Verbs across different possible tags are not very distinguished
Bi-Grams
Filtering Method

62% Bi-Grams

Use Naive Bias scoring on Bi-grams in questions

- Bi-grams produce more unique words
- Bi-grams are more informative on the type of tag overall
Scoring Algorithms

2%  
\[ \text{Tf(class)} - \text{Tf(total)} \]
Take the term frequency per class and subtract by term frequency in total test docs

- Terms are not as unique as expected across tags
- Filler words still determining tag

71%  
\[ \text{Tfidf} \]
Term frequency times inverse document frequency

- Heavily scored infrequent terms
Scoring and filtering (POS + tfidf)

70% POS + tfidf

Applying tfidf to only the nouns in a question

- No overall improvement
- Increase training size
- Tags very similar
Filler words can completely throw off an NLP algorithm, while proper filtering can give surprising improvements.

Scoring words by their uniqueness to the tag can help improve tagging accuracy but comes with challenges.

Filter out by tri-grams
Bi-gram + Tfidf
POS with noun phrases
Questions?
Assessing Toxicity in Wikipedia Comments

Jonathan Innis & Gabriel Britain

Disclaimer: Some comments in this presentation may be offensive to certain viewers. The comments in this presentation do not reflect the opinions of the creators/presenters and are used purely for academic purposes.
Purpose

- Identifying toxicity can prevent users from abusing communication platforms
- Much more efficient than review by human moderators
- Most comments are posted at early hours of the morning (3am) and will be uncaught by human moderators for hours
Nieman Lab is a great website — only an idiot like you would think some other website could possibly be better. You dumb jerk.

First of all, A for effort! But I wasn't a racist like you were, so my grammar is irrelevant (so I'm not a hypocrite, although that's a big word, you should be proud). Also, I should point out that yours didn't improve, so we got nowhere with you. Your spelling makes me inclined to think you're a 'dirty yourself! And I hope at the end that you weren't threatening to kill me. I'll forgive you because you seem cranky, so I'd suggest a nap, you mouth-breathing, stagnant cesspool of human trash.

You're a stupid idiot!
<table>
<thead>
<tr>
<th>id</th>
<th>comment_text</th>
<th>toxic</th>
<th>severe_toxic</th>
<th>obscene</th>
<th>threat</th>
<th>insult</th>
<th>identity_hate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000997932d777bf</td>
<td>Explanation Why the edits made under my username Hardcore Metallica Fan were</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>000103f0d9cb60f</td>
<td>D'aww! He matches this background gojout I'm seemingly stuck with.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>000113f07ec002fd</td>
<td>Hey man, I'm really not trying to edit war. It's just that this guy is con</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0001b41b1e6bb37e</td>
<td>More I can't make any real suggestions on improvement - I wondered if the s</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0001d95ac54c6e35</td>
<td>There appears to be a backlog on articles for review so I guess there n</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>000254654725e87</td>
<td>Congratulations from me as well, use the tools well. · talk “</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0002bcb3da6cb337</td>
<td>BEFORE YOU PISS AROUND ON MY WORK</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00031b1e95af7921</td>
<td>Your vandalism to the Matt Shirvington article has been reverted. Please</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00037261f536c51d</td>
<td>Sorry if the word 'nonsense' was offensive to you. Anyway, I'm not interested in alignment on this subject and which are contrary to those of DuLithgow.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00040093b2667caa</td>
<td>Fair use rationale for Image:Wonju.jpg</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Thanks for uploading Image:Wonju.jpg. I notice the image page specific</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Please go to the image description page and edit it to include a fair use</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>If you have uploaded other fair use media, consider checking that you</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Unspecified source for Image:Wonju.jpg</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Thanks for uploading Image:Wonju.jpg. I noticed that the file's descriptive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>As well as adding the source, please add a proper copyright licensing t</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0005300084f90edc</td>
<td>If you have uploaded other files, consider checking that you have specific</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00054a5e8b50dd4</td>
<td>be a man and lets discuss it-maybe over the phone?</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Frameworks

scikit learn + Keras
Data Inspection

1. Class Distribution
2. Common Toxic Word Inspection
3. Comment Length Inspection
Class Distribution

Number of Examples per Toxic Class

Number of Examples per Class

- Unlabeled
- Toxic
- Severely Toxic
- Obscene
- Threat
- Insult
- Identity Hate
Common Toxic Words
Comment Character Lengths
Random Assignment (based on class frequencies)

<table>
<thead>
<tr>
<th>Class</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toxic</td>
<td>0.10</td>
<td>0.43</td>
<td>0.16</td>
<td>5038</td>
</tr>
<tr>
<td>Severely Toxic</td>
<td>0.01</td>
<td>0.06</td>
<td>0.02</td>
<td>500</td>
</tr>
<tr>
<td>Obscene</td>
<td>0.05</td>
<td>0.23</td>
<td>0.08</td>
<td>2810</td>
</tr>
<tr>
<td>Threat</td>
<td>0.00</td>
<td>0.02</td>
<td>0.01</td>
<td>152</td>
</tr>
<tr>
<td>Insult</td>
<td>0.05</td>
<td>0.23</td>
<td>0.08</td>
<td>2591</td>
</tr>
<tr>
<td>Identity Hate</td>
<td>0.01</td>
<td>0.04</td>
<td>0.01</td>
<td>449</td>
</tr>
<tr>
<td>Micro Avg</td>
<td>0.07</td>
<td>0.30</td>
<td>0.11</td>
<td>11540</td>
</tr>
<tr>
<td>Macro Avg</td>
<td>0.04</td>
<td>0.17</td>
<td>0.06</td>
<td>11540</td>
</tr>
<tr>
<td>Weighted Avg</td>
<td>0.07</td>
<td>0.30</td>
<td>0.11</td>
<td>11540</td>
</tr>
</tbody>
</table>

Baseline
Models

1. Naive Bayes Classifier
2. Support Vector Machines
3. Random Forest Classifier
4. Recurrent Neural Network
Naive Bayes Classifier

- “Bag of Words” model makes sense for toxic comment classification
- Precision, Recall, & F1 strong improvements over baseline

<table>
<thead>
<tr>
<th>Class</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toxic</td>
<td>0.83</td>
<td>0.59</td>
<td>0.69</td>
<td>5042</td>
</tr>
<tr>
<td>Severely Toxic</td>
<td>0.31</td>
<td>0.79</td>
<td>0.44</td>
<td>557</td>
</tr>
<tr>
<td>Obscene</td>
<td>0.78</td>
<td>0.79</td>
<td>0.79</td>
<td>2761</td>
</tr>
<tr>
<td>Threat</td>
<td>0.05</td>
<td>0.78</td>
<td>0.09</td>
<td>163</td>
</tr>
<tr>
<td>Insult</td>
<td>0.65</td>
<td>0.68</td>
<td>0.66</td>
<td>2623</td>
</tr>
<tr>
<td>Identity Hate</td>
<td>0.19</td>
<td>0.58</td>
<td>0.29</td>
<td>481</td>
</tr>
<tr>
<td>Micro Avg</td>
<td>0.53</td>
<td>0.67</td>
<td>0.59</td>
<td>11627</td>
</tr>
<tr>
<td>Macro Avg</td>
<td>0.47</td>
<td>0.70</td>
<td>0.49</td>
<td>11627</td>
</tr>
<tr>
<td>Weighted Avg</td>
<td>0.71</td>
<td>0.67</td>
<td>0.67</td>
<td>11627</td>
</tr>
</tbody>
</table>
Feature Analysis

- Naive Bayes found certain features (unigrams, bigrams, and trigrams) that are most useful to the model
Support Vector Machines

- Word embeddings to produce embeddings for each sentence
- Leveraged GloVe embeddings
- Leveraging custom embeddings could produce better results with greater resources and greater time

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toxic</td>
<td>0.96</td>
<td>0.06</td>
<td>0.12</td>
<td>6090</td>
</tr>
<tr>
<td>Severely Toxic</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>367</td>
</tr>
<tr>
<td>Obscene</td>
<td>0.95</td>
<td>0.09</td>
<td>0.16</td>
<td>3691</td>
</tr>
<tr>
<td>Threat</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>211</td>
</tr>
<tr>
<td>Insult</td>
<td>0.67</td>
<td>0.01</td>
<td>0.03</td>
<td>3427</td>
</tr>
<tr>
<td>Identity Hate</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>712</td>
</tr>
<tr>
<td>Micro Avg</td>
<td>0.93</td>
<td>0.05</td>
<td>0.10</td>
<td>14498</td>
</tr>
<tr>
<td>Macro Avg</td>
<td>0.43</td>
<td>0.03</td>
<td>0.05</td>
<td>14498</td>
</tr>
<tr>
<td>Weighted Avg</td>
<td><strong>0.80</strong></td>
<td><strong>0.05</strong></td>
<td><strong>0.10</strong></td>
<td><strong>14498</strong></td>
</tr>
</tbody>
</table>
Random Forest Classifier

- Resistant to class imbalance
- Decent results that suffered in the macro average performing poorly in the smaller classes
**Recurrent Neural Network (RNN)**

- LSTMs shown to effectively handle long sequence
- Captures sentence structure

<table>
<thead>
<tr>
<th>Category</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toxic</td>
<td>0.57</td>
<td>0.85</td>
<td>0.68</td>
<td>6090</td>
</tr>
<tr>
<td>Severely Toxic</td>
<td>0.34</td>
<td>0.48</td>
<td>0.40</td>
<td>367</td>
</tr>
<tr>
<td>Obscene</td>
<td>0.60</td>
<td>0.80</td>
<td>0.68</td>
<td>3691</td>
</tr>
<tr>
<td>Threat</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>211</td>
</tr>
<tr>
<td>Insult</td>
<td>0.52</td>
<td>0.72</td>
<td>0.61</td>
<td>3427</td>
</tr>
<tr>
<td>Identity Hate</td>
<td>0.67</td>
<td>0.22</td>
<td>0.34</td>
<td>712</td>
</tr>
<tr>
<td>Micro Avg</td>
<td>0.56</td>
<td>0.75</td>
<td>0.64</td>
<td>14498</td>
</tr>
<tr>
<td>Macro Avg</td>
<td>0.45</td>
<td>0.51</td>
<td>0.45</td>
<td>14498</td>
</tr>
<tr>
<td><strong>Weighted Avg</strong></td>
<td><strong>0.56</strong></td>
<td><strong>0.75</strong></td>
<td><strong>0.63</strong></td>
<td><strong>14498</strong></td>
</tr>
</tbody>
</table>
RNN Architecture
Attributions


AGGRO

Declarative Programming in Natural Language

Ryan Beltran
Joseph Gerules
Description & Examples

● Aggro answers questions that are phrased in standard english.

● Examples:
  ○ A year is wild if and only if 2 divides the year evenly. The year is 2018. Is the year a wild one? - TRUE
  ○ A number n is prime if there exists no number m in the range of 1 to n such that m divides n evenly. Is 73 prime? - TRUE
  ○ "If and only if there is rain then there is water. There is not rain. Is there water?" - FALSE
The Five Phases of Aggro

1. Perform Stemming and Text Normalization
2. Perform Lexical Analysis and Tokenization
3. Generate Abstract Syntax Tree
4. Analyze and correlate ambiguous noun phrases
5. Generate, execute, and display Prolog
Stage 1 - Preprocessing

- Utilize Python’s natural language toolkit, NLTK, library for:
  - POS tagging
  - Stemming
  - Remove stop words
  - Lowercasing
  - Tokenize based on POS tagging

Input:
A year is wild if and only if 2 divides the year evenly. The year is 2018. Is the year a wild one?

Output:
a year is wild if and onli if 2 divides the year evenli . the year is 2018. is the year a wild one ?
Stage 2 - Dynamic Tokenization

- Use Lex to parse the now preprocessed input
- Categorize & catch words to assign labels to them
  - Reserved words like “is” or “equals” get tagged as ‘EQUALS’

Input:
a year is wild

Output:
LexToken(A,'a',1,0)
LexToken(UNWORD,'year',1,2)
LexToken(EQUALS,'is',1,7)
LexToken(UNWORD,'wild',1,10)  #UNWORD is short for uniqueword
Stage 3 - Abstract Syntax Tree Generation

- Use yacc to parse the lexemes
  - Words or phrases turn to labeled nodes
  - Rules decide labels and connection order

A year is wild if and only if 2 divides the year evenly.

```
Node: __program__ [17]
  | Node: __rule__ [16]
  |  | Node: __iff then__ [15]
  |  |  | Node: __if__ [13]
  |  |  |  | Node: __is__ [12]
  |  |  |  |  | Node: __modulo__ [9]
  |  |  |  |  | Phrase: { alias:, bound:False } [8]
  |  |  |  |  | Leaf: year [7]
  |  |  |  |  | Node: __numeric const__ [6]
  |  |  |  |  | Leaf: 2 [5]
  |  |  |  |  | Leaf: 0 [10]
  |  |  |  |  | Node: __then__ [14]
  |  |  |  | Node: __is__ [4]
  |  |  |  | Phrase: { alias:, bound:False } [1]
  |  |  |  | Leaf: year [0]
  |  |  |  | Phrase: { alias:, bound:False } [3]
  |  |  |  | Leaf: wild [2]
```

- Considered Approaches:
  - GREMLIN: Levenshtein Optimal Fuzzy Grammars
  - Maximum Entropy Classification: Trained rule based classifier
Stage 4 - Phrase Analysis

- Seeks to connect correlated phrases
- Tasks:
  - Correlate related phrases in an alias table
  - Properly split adjacent noun phrases
  - Label phrases as free or bound
- Phrase correlation based on two part metric:
  - Levenshtein similarity metric
  - Bayesian probability metric
- Split phrases to maximize total similarity
- How do we handle the word “it”?

[Diagram: Link ambiguous phrases]
Stage 5 - Code Generation & Execution

- Use the AST’s labels & node structure to write generic Prolog functions.
  - Add id’s to each label to ensure uniqueness of generically named functions
  - Push all generated rules into the query statement to create a scope
- Use SWIPL to call SWI Prolog from Python.

```
water is wet . is water wet ?
is_10( A, A ).  // water is wet - Question form
is_4( A, A ).   // water is wet - Rule Form

query_11(  ) :- is_4( Phrase_0, Phrase_1 ), is_10( Phrase_0, Phrase_1 ). /*11*/ // Adding both rules creates a scope

query_11(  ) is:
true
```

Handled in Python to allow more usage

Uses the payloads from the AST
Future Development

● Produce more grammar rules for a more robust system.
  ○ This would allow for more edge cases to be handled
  ○ Different styles of questions could be added
● Integrate pronoun and ambiguous word binding more thoroughly.
  ○ Look for nearby nouns.
● Make the outputted code more readable
  ○ Formatting
  ○ Implement using attributed objects instead of rules
● Fuzzy grammars to handle oddly worded input
● Improved phrase splitting
  ○ Is the grey cat very large?
    ■ Is the grey | cat very large  or  Is the grey cat | very large  or  Is the grey cat very | large
● Improved handling of free variables
  ○ The year y is a leap year if it is divisible by 4.
    ■ Y isn’t a specific year. It is an unbounded free variable.
Math Question Answering
(SemEval Task 10)

Kevin Sittser
The Problem

- **Closed-vocabulary algebra**, e.g. "Suppose $3x + y = 15$, where $x$ is a positive integer. What is the difference between the largest possible value of $y$ and the smallest possible value of $x$, assuming that $y$ is also a positive integer?"
- **Open-vocabulary algebra**, e.g. "At a basketball tournament involving 8 teams, each team played 4 games with each of the other teams. How many games were played at this tournament?"
- **Geometry**, e.g. "The lengths of two sides of a triangle are $(x-2)$ and $(x+2)$, where $x > 2$. Which of the following ranges includes all and only the possible values of the third side $y$?"

- Success metric: Percentage of problems solved
My Approach

- Ignore diagrams
- Equation solver!
- Dimensional analysis??
- Write a parser??
- Ignore wordy problems
- SymPy
- Algorithms??
Program Structure

- Question parser: Look for equations
- Equation parser: Convert to SymPy-readable (if possible)
- Solver: Equations, expressions → Result!
- Find closest valid answer
- (If can’t solve problem, output “C” (or “5”))
Complications

- Training?
- SymPy `solve()` output
- SymPy can’t handle everything
- So many equation formats
Results

- 20.82% correct answers!
- But a random guesser solves 19.38%
- Not very good
- Only target problems: 22.70% (vs. 19.03%) → a little better
Potential Improvements

- Training program?
- Write my own labels?
- Make sure all equations are LaTeX
- Accept more operation types (SymPy research)
- Syntactical analysis??
- Long, long process
Questions?