**COMP90049 Knowledge Technologies Project 2**

|  |
| --- |
| **Twitter Trolls and the Tweeters who Love them** |

**1. Introduction**

To some extent, the birth of the Internet has inexplicably changed the way we humans communicate. This can be observed with the ever so blossoming spectrum of different communication platforms. First came email, blogs, forums, then social media, followed closely by instant messaging, as well as the more recent uprising of live stream chatting [CITATION NEEDED].

However, akin to the sprouting of the methods of communicating, new types of communicators have evolved along, with some of them erring to the negative side. One such are termed ‘Internet Trolls’, which describes an individual posting comments or some other form of online content, with the intention of attacking, offending, disrupting, or to cause trouble within the community [https://www.lifewire.com/types-of-internet-trolls-3485894].

This project focuses on ‘trolls’ originating from one of the more popular social media platforms, Twitter. With the help of Machine Learning methodologies, this project attempts to analyse and justify whether tweet data is useful in identifying ‘troll’ tweets by Twitter users associated with the Internet Research Agency [specification].

**2. Knowledge Hypothesis**

Trolls are often frowned upon, as their negativity often creates chaos amongst the community. It would be beneficial for social media sites to apply methodologies such as machine learning to identify and filter out troll tweets. This would help create a better, safer platform for users to communicate.

Although it may be plausible to hypothesise that a troll tweeter is likely to post additional troll tweets, the nature of troll identification lies in the tweet itself. As in, filter mechanisms would generally prefer to catch these tweets immediately as they are posted, not after a specified time, even if the user behind it is inherently not a troll.

**3. Domain and Sources**

The dataset used is entitled ‘Troll factories’, a corpus of 3 million Russian troll tweets tweeted by just under 3000 unique users, released for public use (Linvill, Darren and Patrick Warren (2018)).

Due to the very nature of the dataset, which primarily contains Russian language, we will be using a subset of the data, specifically three datasets encompassing approximately 223k tweets from 175 randomly chosen users, entitled ‘large’, ‘medium’, and ‘small’ respectively.

These datasets, preprocessed by academic staff at The University of Melbourne, differ in the number of English terms extracted from the original document. There are two versions of the data, each containing the top English terms according to document frequency, as well as terms with the greatest Chi-Square and Mutual Information values. These datasets have been further partitioned into train, development, and test sets, with a ratio of roughly 60%, 20%, and 20% respectively.

|  |  |  |
| --- | --- | --- |
| Dataset | Tweets | Features |
| train-best200.csv | 122637 | 200 |
| dev-best200.csv | 56194 | 200 |
| train-best50.csv | 122637 | 50 |
| dev-best50.csv | 56194 | 50 |
| train-best10.csv | 122637 | 10 |
| dev-best10.csv | 56194 | 10 |

Table 1: Number of tweets and features

**3.1. Differences**

A significant difference the processed datasets possess in comparison to the raw data, is that non-English characters have been removed, and the remaining had their cases folded. Hyperlinks and hashtags have also been stripped out.

**3.2. Terminology**

The following lists the descriptions of the class labels present in the dataset [CITATION]:

|  |  |
| --- | --- |
| Class | Description |
| RightTroll | Broadcast nativist and right-learning populist messages |
| LeftTroll | Sent socially liberal messages, with an overwhelming focus on cultural identity |
| Other | An amalgamation of several other categories of trolls, including “HashtagGamer”, “Commercial”, “NewsFeed”, and others. |

Table: Descriptions of class labels

With the ‘Other’ class, this implies that the tweet is from someone who might reasonably have been a right/left troll, but actually was not, although they might still have been some other kind of troll.

**3.4. Limitations**

As the raw tweet data is primarily written in Russian, it may be somewhat subjective to only evaluate trolls based solely on English text, as it is possible that troll tweets can also be written in Russian. Further, the fact that the top X words chosen as features may not necessarily represent all the words given in the data, although this would help to avoid overfitting the dataset.

Another observation would be that the left and right troll refer to specific groups of tweeters who are centered around an election consisting of a set number of candidates from different parties. In future elections however, these candidates, or even parties, may very well change.

**4. Problem**

Seeing as there are three classes of trolls labelled in the data, this implies a multiclass classification problem. Binary classifiers would require a slight change in approach to be applied to multiclass domains. In the case of scikit-learn, classifiers like SVM can have the ‘multi\_class’ parameter set to achieve this.

**4.1. Methodology**

The system leverages a couple of supervised machine learning algorithms, using the Python scikit-learn library [CITATION]. Given that the labels have been provided, the algorithms are first fitted using the training data and evaluated using the development dataset. An outline and short description of the different algorithms are described below.

**4.2. Decision Tree**

Edit Distance is a widely used distance metric designed to determine similarity between two strings. The system uses three different variants, namely global edit distance, local edit distance, and Damerau-Levenshtein distance.

The basic idea behind global edit distance is transforming an input string into each dictionary entry based on scores associated with insert, delete, replace, and match operations (Langmead, 2018).

Local edit distance is based around the idea of finding the longest matching substring of the word.

**4.3. Random Forest**

The concept of neighbourhood search is somewhat akin to edit distance, in which the algorithm generates all variants of a string s that utilise at most k changes, be it insertions, deletions, or replacements (Nicholson et al. 2017).

However, since the dictionary is known, we can, given a parameter, search the dictionary via running the Needleman-Wunsch algorithm with Levenshtein Distance parameters, and returning words that match the score.

**5. Results**

The performance of the system was measured using accuracy, which is given by the number of correct predictions the classifier made on the ‘dev’ dataset, after being trained on the ‘train’ dataset.

|  |  |  |  |
| --- | --- | --- | --- |
| Classifier | Parameter | Dataset | Accuracy |
| Decision Tree | Gini  1,1,1,1 | large  best | 66.46% |
| Decision Tree | Entropy  1,1,1,2 | large  best | 66.22% |
| Random Forest | None  1,1,7 | large  best | 69.93% |
| Naïve Bayes | Gaussian  1,1,5,1 | large  best | 70.43% |
| Naïve Bayes | Bernoulli  1,1,5,2 | large  best | 70.33% |
| Logistic Regression | 1,2,2 | large  best | 70.20% |
| SVM | 1,1,8 | large  best | 69.68% |

Table 2: Overall Comparison

**5.3. Comparing Datasets**

The N-Gram algorithm has also been run on the Birkbeck dataset, with N = 2. Results are shown below, comparing both datasets. The algorithm performs much more poorly on written misspellings.

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | Accuracy | Precision | Recall |
| Wikipedia | 55.47% | 68.06% | 81.50% |
| Birkbeck | 19.17% | 7.43% | 37.16% |

Table 2: Dataset Comparison

Graph 1: Overall Comparison

Graph 2: Accuracy Comparison

Graph 3: Precision Comparison

Graph 4: Recall Comparison

**6. Discussion**

The results show that, in terms of precision and recall, Damerau-Levenshtein outperforms the rest. Accuracy is rather similar, with the exception of local edit distance.

**6.1. Neighbourhood Search**

|  |  |
| --- | --- |
| Misspelled | Correct |
| 'appereance' | 'appearance' |
| 'catapillers' | 'caterpillars' |
| 'dissapears' | 'disappears' |
| 'exculsivly' | 'exclusively' |
| 'governmnet' | 'government' |
| 'harrasment' | 'harassment' |
| 'moleclues' | 'molecules' |
| 'parituclar' | 'particular' |
| 'resssurecting' | 'resurrecting' |
| 'startegic' | 'strategic' |
| 'unfortunatley' | 'unfortunately' |

Table 4: Sample of misspelled words not detected by neighbourhood of 1

While at first glance, even though the misspelled words look almost identical to the correct word, they went undetected due to being just out of the boundary.

Increasing the neighbourhood to higher values may detect the error, but also produces many more results, hence lowering the precision.

Neighbourhood search has the advantage that it is not affected by the fact that some misspelled tokens are present in the dictionary, given that a neighbourhood of > 0 is specified.

This method has one major pitfall, in which it might not return any matches, especially if the dictionary is small, or if words are exceptionally longer than the rest.

**6.1. N-Grams**

N-grams however, is more consistent, as it does not rely on having a word present in the dictionary. It is less sensitive to how strings are ordered, as matches can be anywhere, but is more sensitive to longer substrings.

In terms of typographical errors, n-grams vaguely demonstrates that people do think out loud while typing on the keyboard as each n-gram somewhat mimics the sound produced while ‘breaking-down’ words into smaller units of pronunciation.

**6.3. Edit Distance**

Local edit distance performs the most poorly here, mainly due to the fact that the dictionary is relatively large, as well as having a relatively small alphabet size of 26. The bigger the number of words, the more likely that longer substrings are to match a particular string. This causes the algorithm to return a much larger hence the low precision score.

Based off the Needleman Wunsch algorithm, both Levenshtein distance and Damerau-Levenshtein distance perform well on the dataset. Damerau-Levenshtein is an improvement over Levenshtein, whereby it also allows for transposition of two adjacent characters (Setiadi, 2018).

This would consider misspellings that could be corrected with at most one edit operation, hence the higher overall score compared to Levenshtein.

Mispelled: acquiantence Correct: acquaintance

|  |  |
| --- | --- |
| Algorithm | Matches Returned |
| Levenshtein | acquaintance,  acquiescence,  acquiesence,  acquittance |
| Damerau-Levenshtein | acquaintance |

Mispelled: dimensional Correct: dimensional

|  |  |
| --- | --- |
| Algorithm | Matches Returned |
| Levenshtein | digestional,  dimensional |
| Damerau-Levenshtein | dimensional |

Mispelled: directly Correct: directly

|  |  |
| --- | --- |
| Algorithm | Matches Returned |
| Levenshtein | dejectly,  directly,  erectly,  oriently,  priestly |
| Damerau-Levenshtein | directly |

Tables 5: Sample of words Damerau-Levenshtein is able to correct, but not Levenshtein

**7. Improvements**

There are many possible ways to improve on the methodology used in this project. One such is incorporating state-of-the-art deep learning methods, such as Convoluted Neural Networks. These methods have the ability to incrementally update its knowledge about troll users over time, with each new dataset fed in.

Incorporation of an ensemble method such as bagging (Hauskrecht, 2018), which would, given all the results from the different algorithms, choose the best match based on majority vote. This is akin to a Random Forest classifier, which has the potential to be improved via boosting.

Another way would be to implement a ‘keyboard-distance’ algorithm that would determine the relative distance of each key to every other key, and base word distances on this. This might improve the typographical suggestions but may be tied to a specific keyboard layout such as QWERTY.

**8. Conclusions**

In all, while this analysis may be far from optimal, it suggests that there are many different approaches and methods to determine the types and causes of typographical errors.

However, the algorithms chosen might not be close enough to be relevant in justifying the hypotheses stated initially. More advanced methods are required to further test the hypotheses, strengthening the consensus that spelling correction is still considered a knowledge task, even with the many advancements in AI and machine learning,

**References**

Linvill, Darren and Patrick Warren (2018) Troll factories: The Internet Research Agency and state-sponsored agenda building (working paper). Clemson University.

Roeder, Oliver (2018) Why we’re sharing 3 million Russian troll tweets. In FiveThirtyEight. 31 Jul 2018, https://fivethirtyeight.com/features/why-were-sharing-3million-russian-troll-tweets/.

Roeder, O. (2018). 3 million Russian troll tweets. [online] GitHub. Available at: https://github.com/fivethirtyeight/russian-troll-tweets/.

Project 2: Twitter Trolls and the Tweeters who Love them. (2018). [ebook] School of Computing and Information Systems, The University of Melbourn. Available at: https://app.lms.unimelb.edu.au/bbcswebdav/pid-7024325-dt-content-rid-47549113\_2/courses/COMP90049\_2018\_SM2/projects/2018S2-90049P2-spec.pdf.

Langmead, B. (2018). Dynamic programming and edit distance [Ebook] (p. 10). Johns Hopkins Whiting School of Engineering: Department of Computer Science. Retrieved from http://www.cs.jhu.edu/~langmea/resources/lecture\_notes/dp\_and\_edit\_dist.pdf

Hauskrecht, M. (2018). CS 2750 Machine Learning Lecture 23: Ensemble methods. Bagging and Boosting [Ebook]. University of Pittsburgh. Retrieved from https://people.cs.pitt.edu/~milos/courses/cs2750-Spring04/lectures/class23.pdf

Setiadi, I. (2018). Damerau-Levenshtein Algorithm and Bayes Theorem for Spell Checker Optimization. Retrieved from http://file:///C:/Users/Xenzaki/Downloads/Damerau-LevenshteinAlgorithmandBayesTheoremforSpellCheckerOptimization.pdf