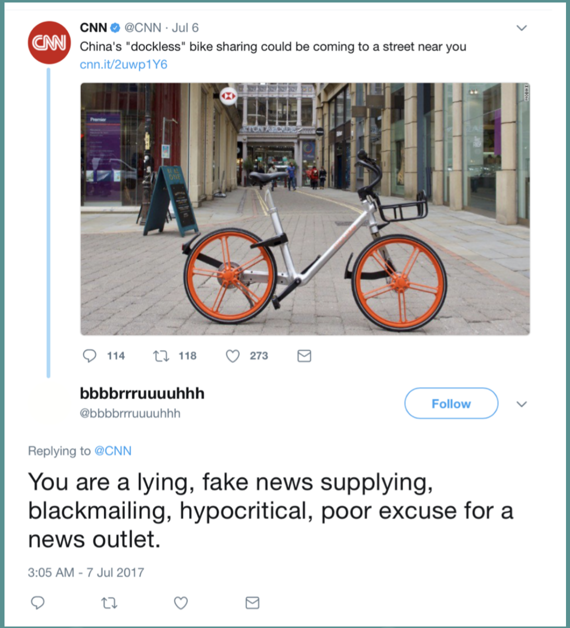
WikiPedia Online Attacking Detection

Xing Zhao && Sirui Li

Introduction & Motivation

- Social media platforms have become rife with undesird effects, especially personal attacking.
- Offensive comments impair user experience and reduce potential customs
- Traditional approaches, such as crowd-workers, is time consuming and expensive.
- Machine learning method can handle data at large scale with high accuracy.



Brief Literature Review

- Our work based on Wulczyn’s work (2017).
- N-gram is the approach that they applied on data preprocessing
- Logistic regression(LR) and Multi-Layer Perceptrons(MLP) are two basic models implemented by the author

Motivation of Our Project

- Can we use other models, such as SVM and Random Forest? How do they perform?
- Can we use some optimal features rather than N-gram? How do such features work?

Problem Solution

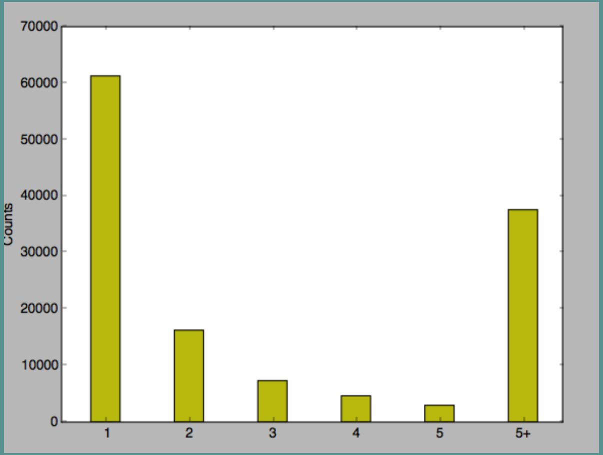
Given a global set of comments, we aim to dectect the personal attacking comments by

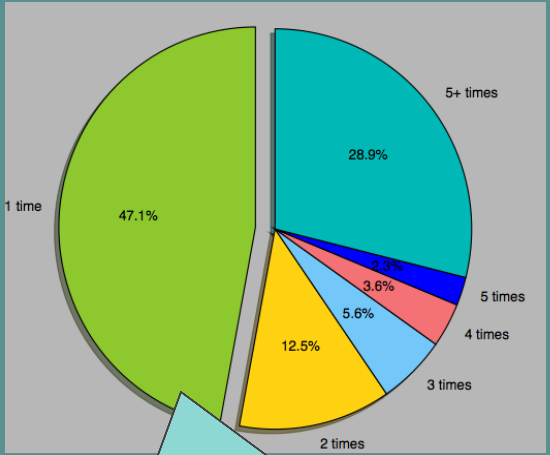
1. Data Processing:
   a. N-gram (word level) features (Wulczyn et al. 2017)
   b. Advanced NLP features, such as sentiment, structure, topic features, etc. (Optimizing)
2. Model Designing:
   a. Logistic Regression(LR) and Multi-Layer Perceptrons(MLP) (Wulczyn et al. 2017)
   b. Random Forest and SVM (Extra Work)

Data Set Overview

- This dataset collects over 100k annotated discussion comments from Wikipedia in English.
- Every comment has been labeled by around 10 annotators on whether it is a personal attack or not.
- In summary, there are around 14,000 comments annotated as personal attacks out of 115,864 comments in total.

Approach 1: Using N-gram Features






```
key:ultimatley     ,word:ultimatley
key:solout     ,word:soloution
key:gayyyyyyyyyy     ,word:gayyyyyyyyyy
key:privlig     ,word:privlige
key:dofficult     ,word:dofficult
key:probabil     ,word:probabillity
```

Words that appear only 1 time are typos.
Drop them! However, still a big matrix with
~110,000 * 40,000

Approach 1: Using N-gram Features

Such Matrix are extremely sparse!!

Preprocessing	Feature Size	Memory Size
Original N-gram	160,000	~136 GB
Drop words appeared only 1 time	40,000	~34 GB
Drop words appeared <=2 times	26,000	~23 GB

We have low confidence to drop such words.

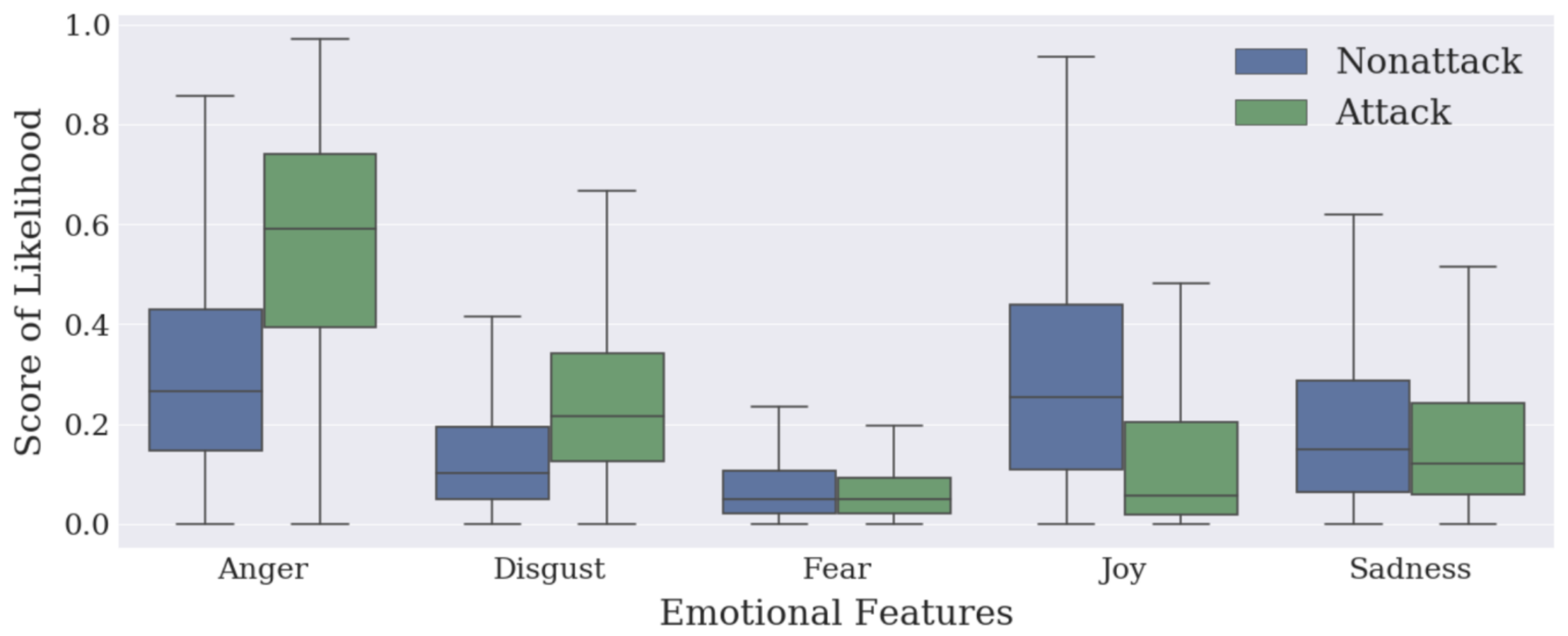
That means, the data matrix will be 110,000 * 160,000!

Approach 2: Using Advanced NLP Features

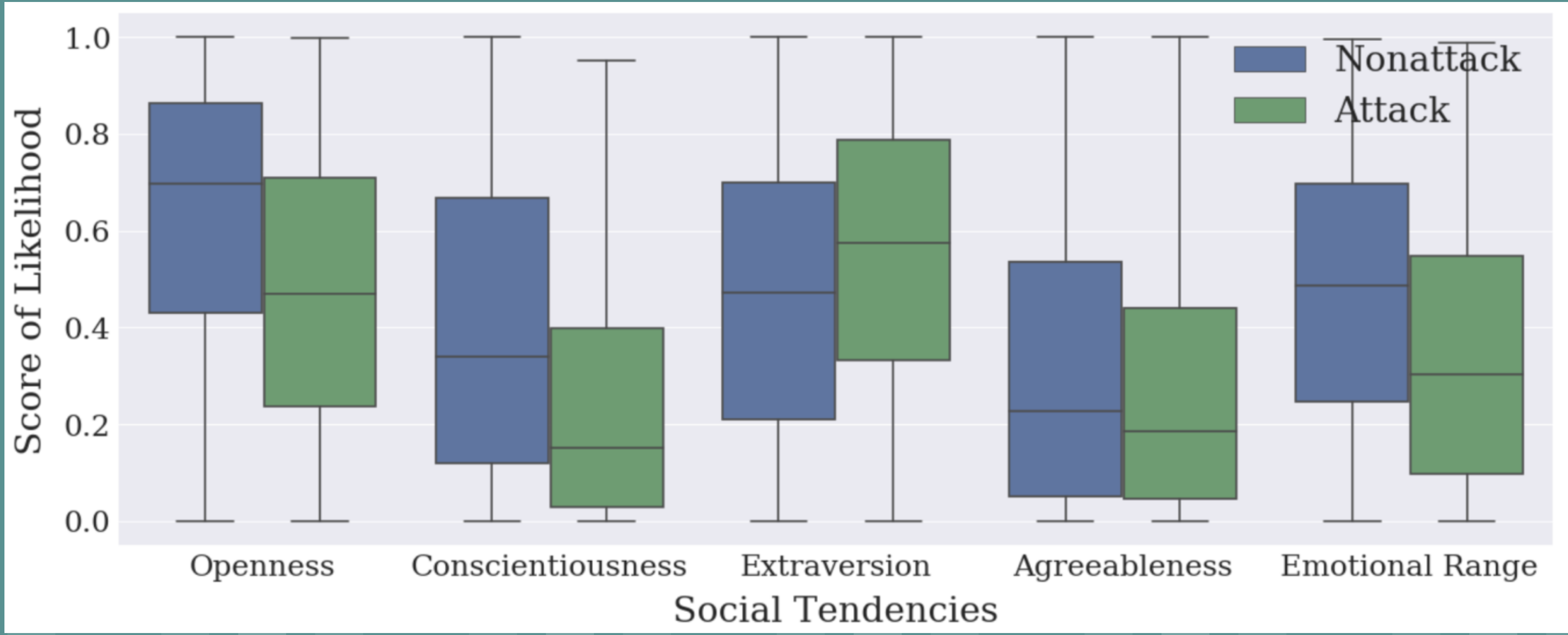
Feature Name	Abbr	Feature Size
Language Structure	LS	100
Communication Sentiment	CS	14
Content Relevance	CR	25
Latent Topic	LT	20
Total (LS+CS+CR + LT)	all	159

Much Smaller than
N-gram!!!

Sentiment Analysis: Emotions



Sentiment Analysis: Social Tendencies



Classification Performance

1. Method using N-gram (Wululczyn, 2017)

The new model is good here!

	Logistic Regreation	Multi-Layer Perceptron	Random Forest
AUC Score (Std)	0.9145 (0.0050)	0.8915 (0.0076)	0.9206 (0.0052)

Classification Performance
2. Method using Advanced NLP Features

	Log-Reg	SVM	RF	MLP
CS	0.8859	0.8752	0.8851	0.8953
LS	0.7120	0.6022	0.7178	0.7514
CR	0.9052	0.8982	0.8895	0.8875
LT	0.6277	0.6031	0.7997	0.6494
LS-CS	0.8389	0.8465	0.8943	0.8989
LS-CS-CR	0.8657	0.9038	0.9215	0.9330
LS-CR-LT	0.8034	0.8207	0.8799	0.8942
all	0.8646	0.9063	0.9244	**0.9356**

Comparison

	Feature used	Highest AUC Score	Required Memory Size
Method 1 (drop 2 times word)	N-gram	$\boxed{0.9206}$	$\boxed{36\ GB}$
Method 2 (our)	Advanced NLP	0.9356	0.34 GB

This shows that the new method saves almost 100 times memory and is more precise!!

Conclusions

1. We implemented Wulczyn's experiment, and tried one more classification model, Random Forest, which performs better than the two models used in their papers.
2. We designed a new approach for detecting personal attacking online using some advanced NLP features, such as language structure, communication sentiment, etc.
3. Our new approach gained better classification performance (AUC score) than original N-gram approach.
4. Our new approach needed much less memory size. Input matrix scales dramatically and significantly decline from ~110000*40000 to ~110000*159.

Future Work

Can we find an suitable method to translate typos into correct forms? It is another way to reduce the size of N-gram features.

Thank you!