Detecting Toxic Comment

Case of Wikipedia Discussions

Ang Li, ANL125@pitt.edu

Motivation

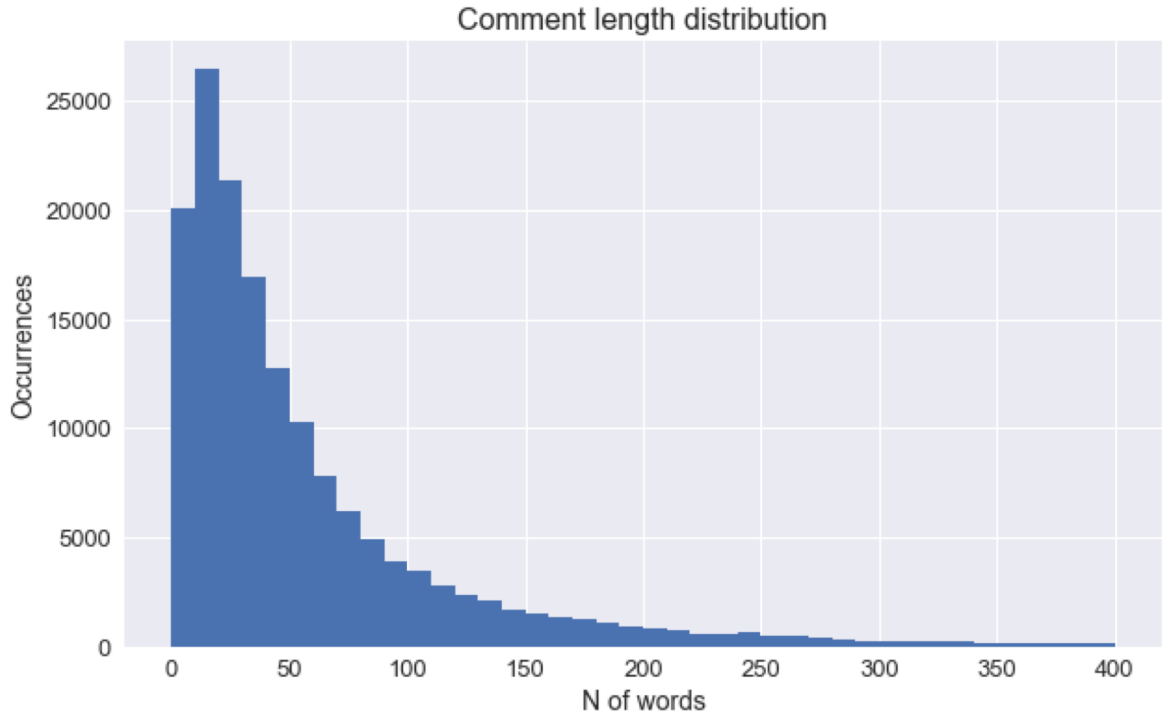
- **Social interaction and discussion** are the core mechanisms for most online communities

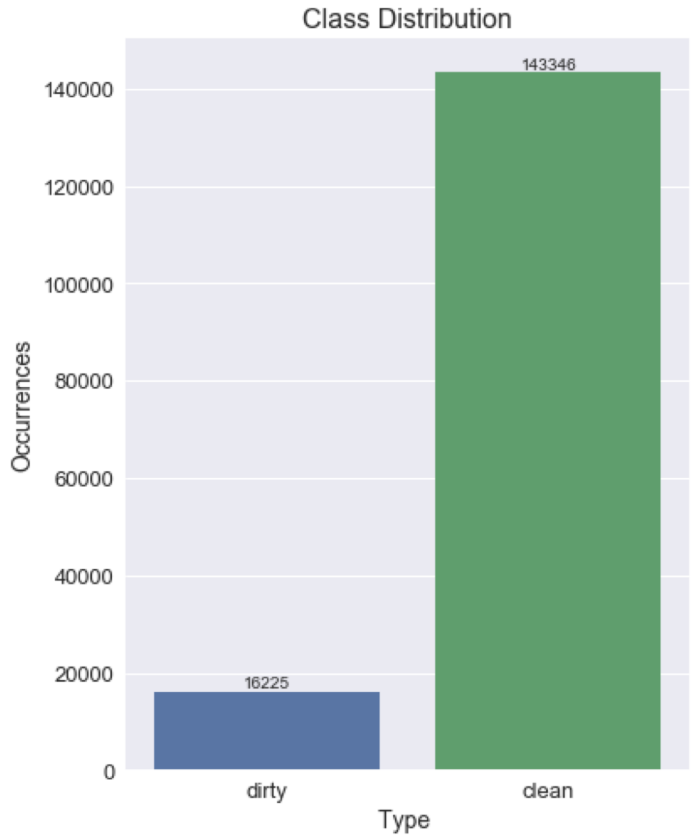
- The ***anonymity*** afforded by such online communities has led to the increase of **misbehavior**
  - E.g. abuse and harassment, spread of propaganda, hate speech, trolling, etc.
- Impede the healthiness of the online environment

- **Project Goal**: provide a technical tool to detect the toxic social interactions accurately and effectively

Data

- 159,571 total Wikipedia discussions with human labeled into “dirty” comments or not





Project Goal: Detect the “dirty” comments -- 10% of the dataset

Method

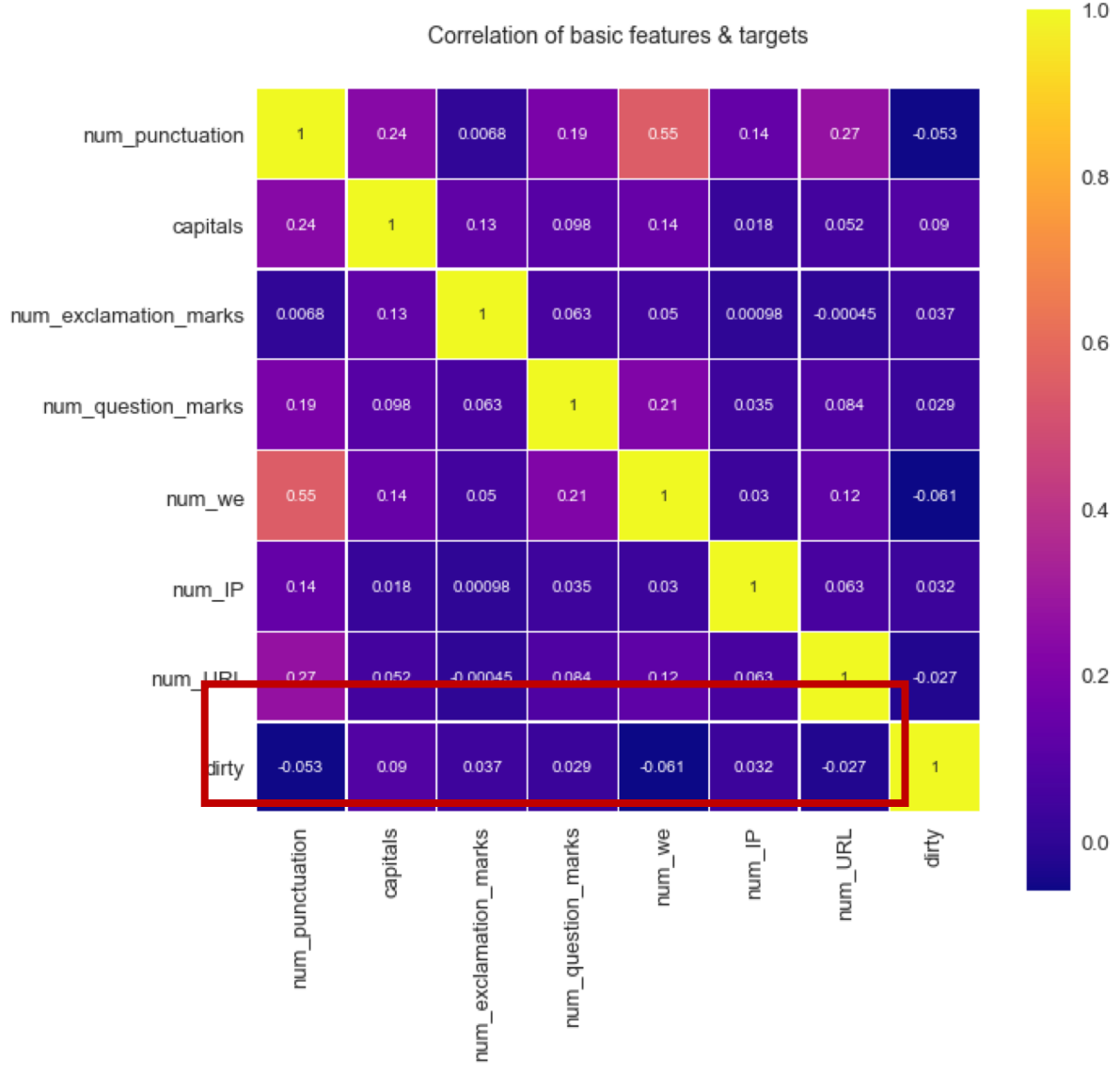
- Feature engineering
  - **Language features**(14) extracted from raw comments
  - **A: Lexicon features** (14 + 3) with two additional dictionaries on positive/negative/semantic words and scores
  - **B: Word Embedding** (14 + 3 + 200) with word2vec features

- Prediction
  - Baseline model:
    - Logistic Regression (L2), Naïve Bayes, Decision Tree (Gini, Entropy)
  - Neural Network (8 hidden layers * 100 hidden units)
  - Ensemble model (Bagging and Ada-boost on 100 DT models)
  - Performance comparison on Feature set A and B

- Evaluation:
  - Overall accuracy & F1-score

Feature engineering – Language features

- Language Feature set from raw comments (17)
  - **Length, # words, # punctuation**
  - **# Capitals, # exclamation marks, question marks, other symbols** – *usually used by toxic comments*
  - **# We, # simile faces** – usually a good sign
  - **IP** – non-registered users are more likely to be associated with toxic
  - **# URL** -- *used for fact updating in comments*



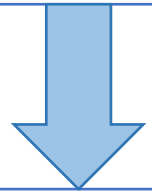
Lexicon features

- Raw comment cleaning
  - Remove URL/IP/number/smiles/symbols
  - Remove stop words
  - Lemmatization

- Feature expansion from semantic dictionary
  - # positive, negative words used
  - Positive semantic score
    - SemEval-2015 English Twitter Lexicon

Raw comments

"Explanation
Why the edits made under my username Hardcore Metallica Fan were reverted?
Check the following websites:
www.superbeyin.com/sohbet/sohbet.htm. 89.205.38.27"

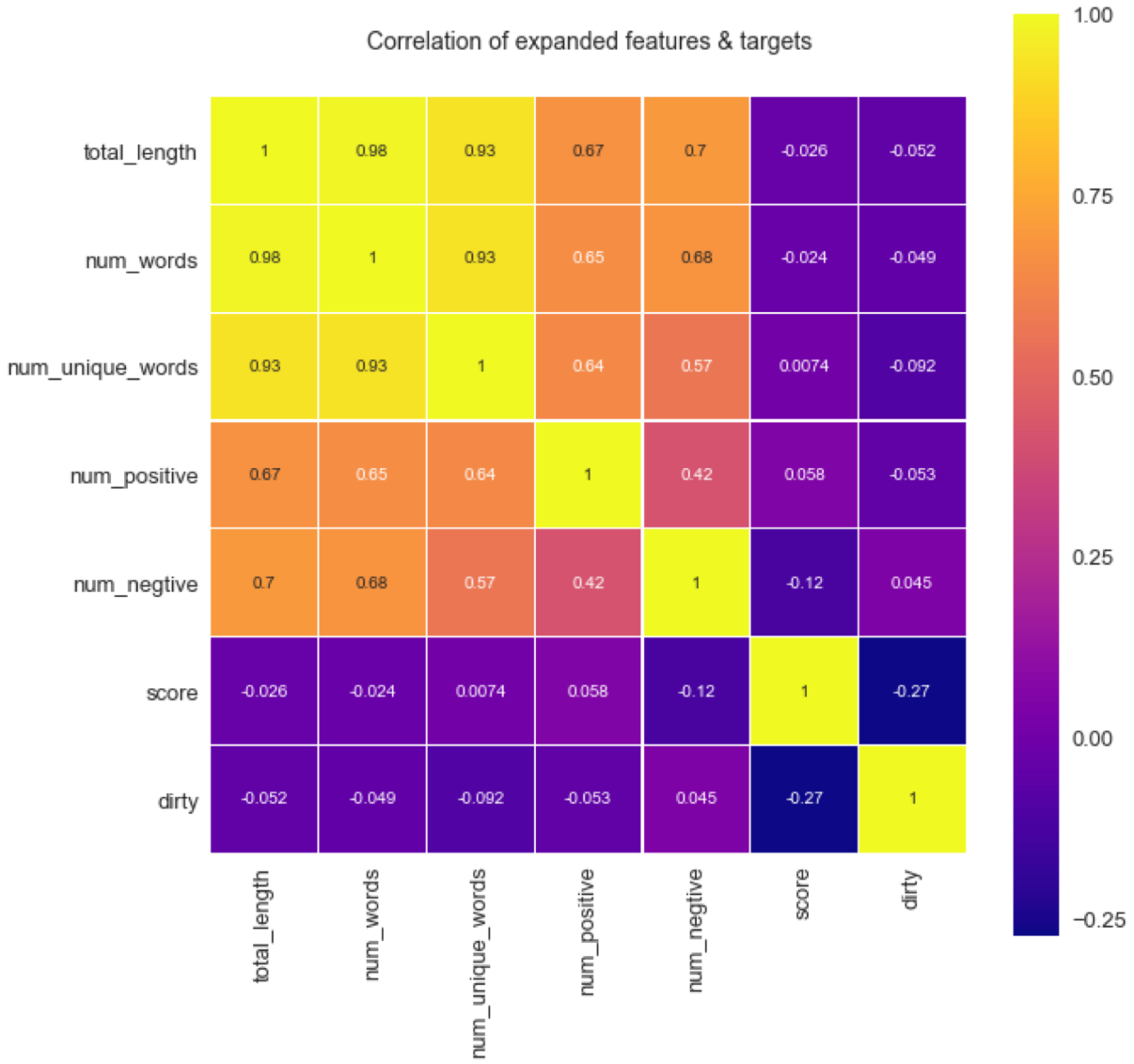


Clean comments

'explanation why edit make username hardcore metallica fan reverted check follow website'

Lexicon features

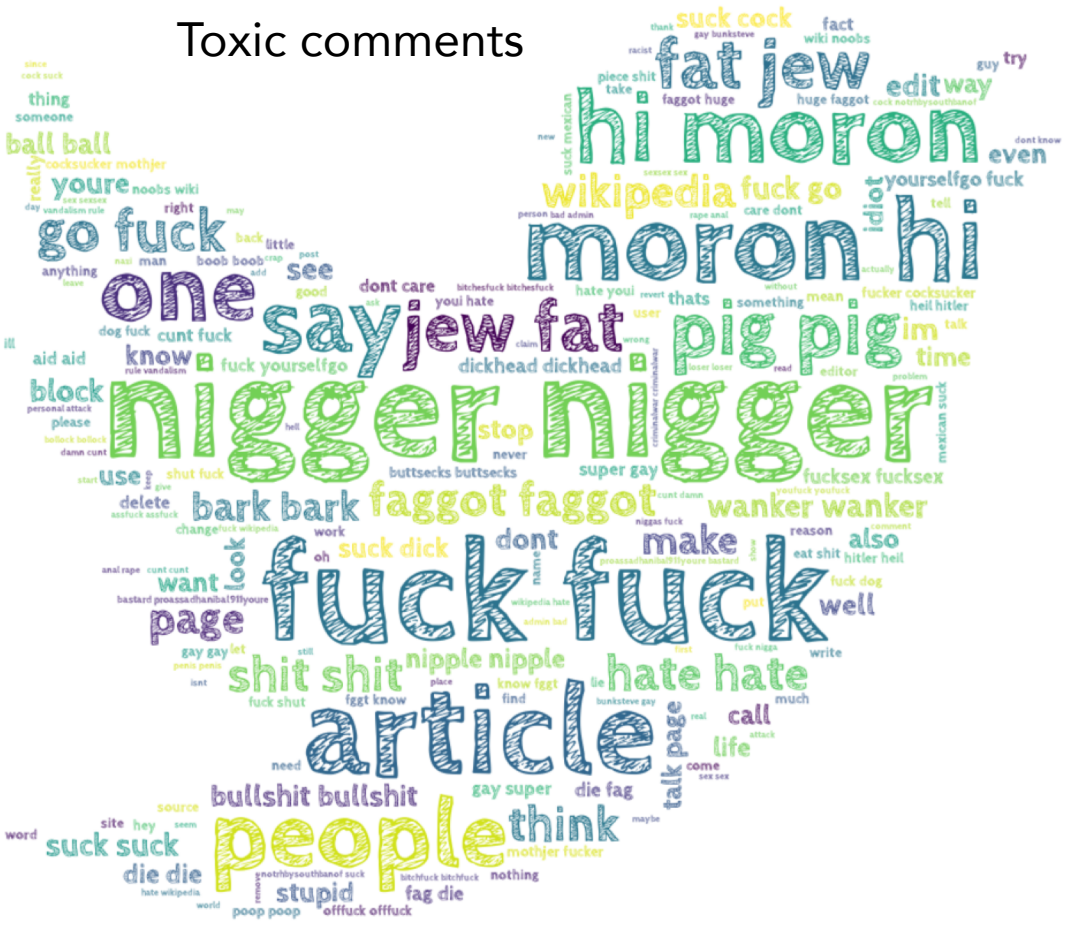
- Raw comment cleaning
  - Remove URL/IP/number/smiles/symbols
  - Remove stop words
  - Lemmatization
- Feature expansion from semantic dictionary
  - # positive, negative words used
  - Positive semantic score
    - SemEval-2015 English Twitter Lexicon



Word cloud comparison

Clean comments

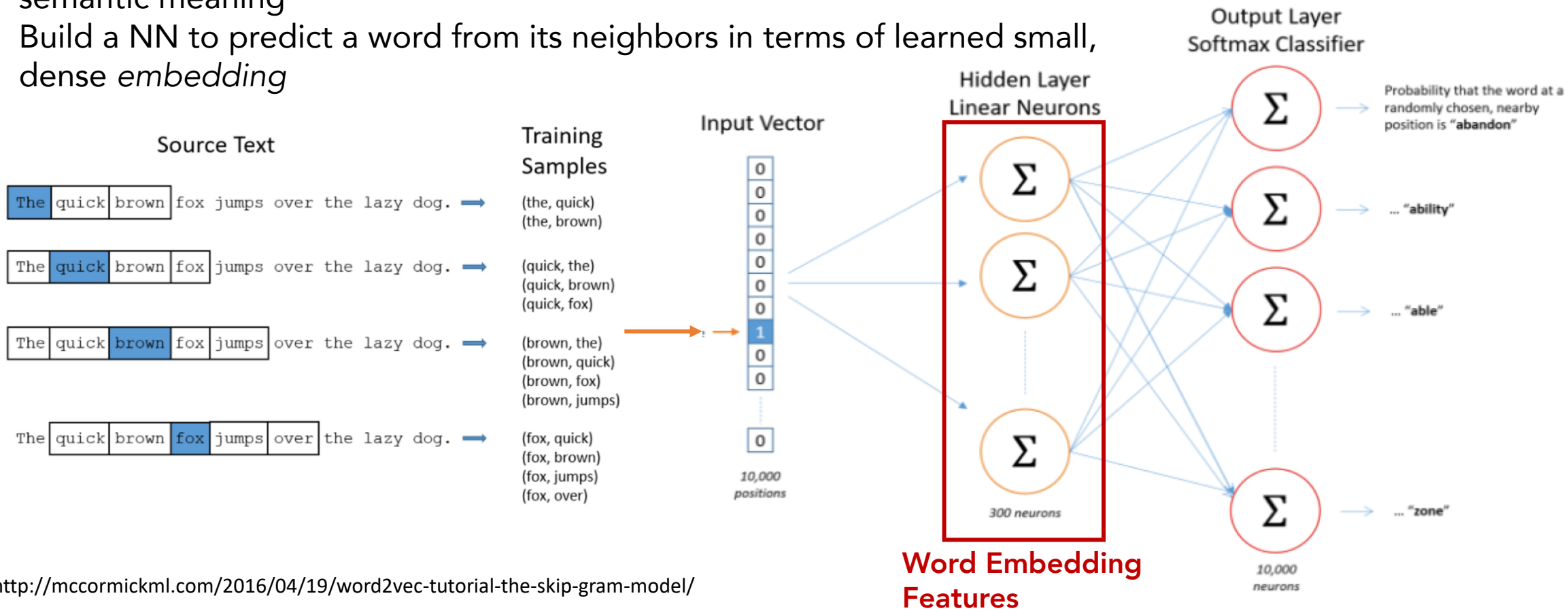
Toxic comments



Word Embedding

- Map a high dimensional one-hot encoding of words to a lower dimensional vector
- Maintain word context and its meaning to some extent. E.g. V(king) – V(man) + V(woman) = V(queen)

- Utilized the **skip-gram** model proposed by *Mikolov et al.*
- **Basic idea:** words that appear in the same contexts (nearby words) share semantic meaning
- Build a NN to predict a word from its neighbors in terms of learned small, dense *embedding*



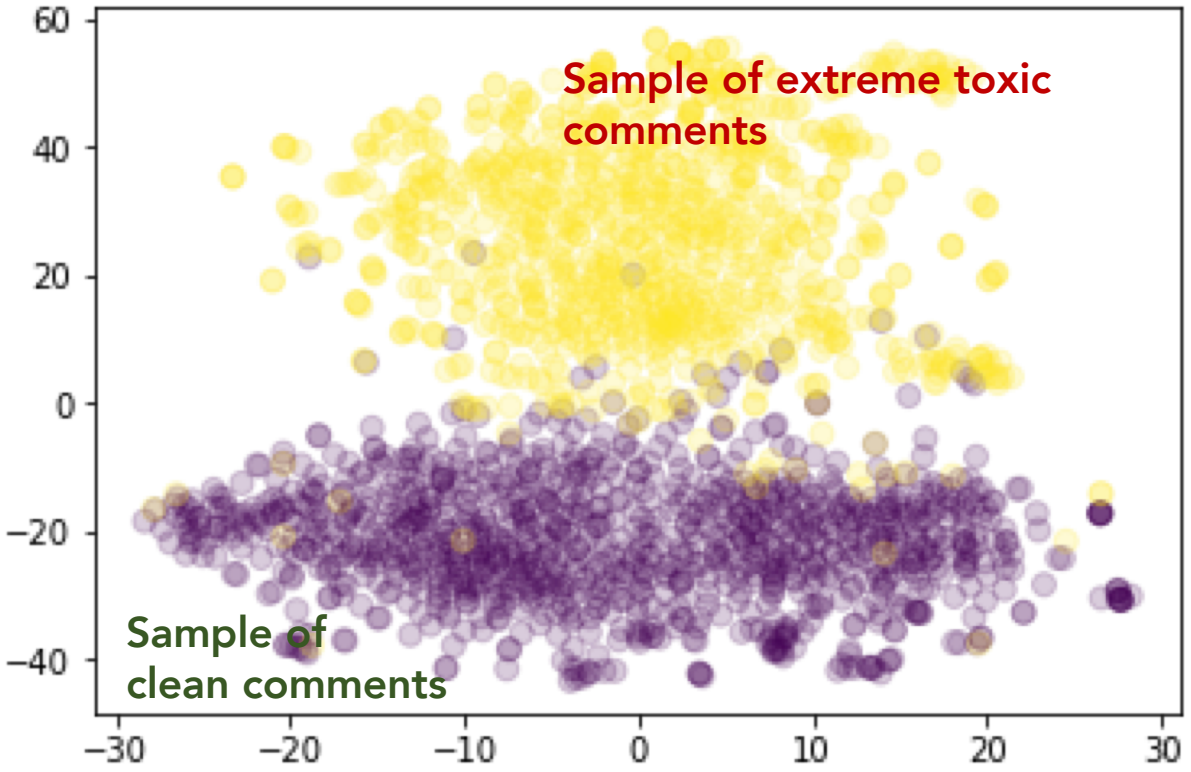
Source: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Word Embedding Feature

- Window size = 10
- Hidden layer units = 200

- For each word: learn a 200 dimension vector
  - Closest words for ***revert*:**
  - *reverter, undoing, vandalistic …*

- For each comment: average over all the word vectors



t-SNE of Sample comments visualization

Detection Results

		Feature set A: Language + Lexicon		Feature set B: Language + lexicon + W2V	
		Accuracy	**F1 score**	**Accuracy**	**F1 score**
Base-line Model	Logistic Regression	Train: 0.90 Test: 0.90	Train: 0.13 Test: 0.13	Train: 0.93 Test: 0.93	*Train: 0.53* *Test: 0.52*
	Naïve Bayes	Train: 0.89 Test: 0.89	Train: 0.24 Test: 0.23	Train: 0.91 Test: 0.91	*Train: 0.605* *Test: 0.606*
	Decision Tree	Train: 0.96 Test: 0.87	Train: 0.78 Test: 0.36	Train: 0.98 Test: 0.91	*Train: 0.93* *Test: 0.58*
Ensemble base=DT	Bagging	Train: 0.99 Test: 0.91	Train: 0.99 Test: 0.42	Train: 0.99 Test: 0.95	*Train: 0.99* *Test: 0.68*
	Ada-boost	Train: 1 Test: 0.90	Train: 0.99 Test: 0.38	Train: 1.0 Test: 0.91	*Train: 1.0* *Test: 0.57*
Neural Network (100 * 8)		Train: 0.92 Test: 0.91	**Train: 0.51** **Test: 0.42**	*Train: 0.99* *Test: 0.99*	***Train: 0.99*** ***Test: 0.72***

Conclusion and Future work

- Word embedding features improve the model performance dramatically
- Neural network performed the best compared with other techniques

- Some sampling techniques to deal with the high unbalanced data from two classes
- NN + Ensemble method may perform even better but need long time to run

Questions? Thanks!